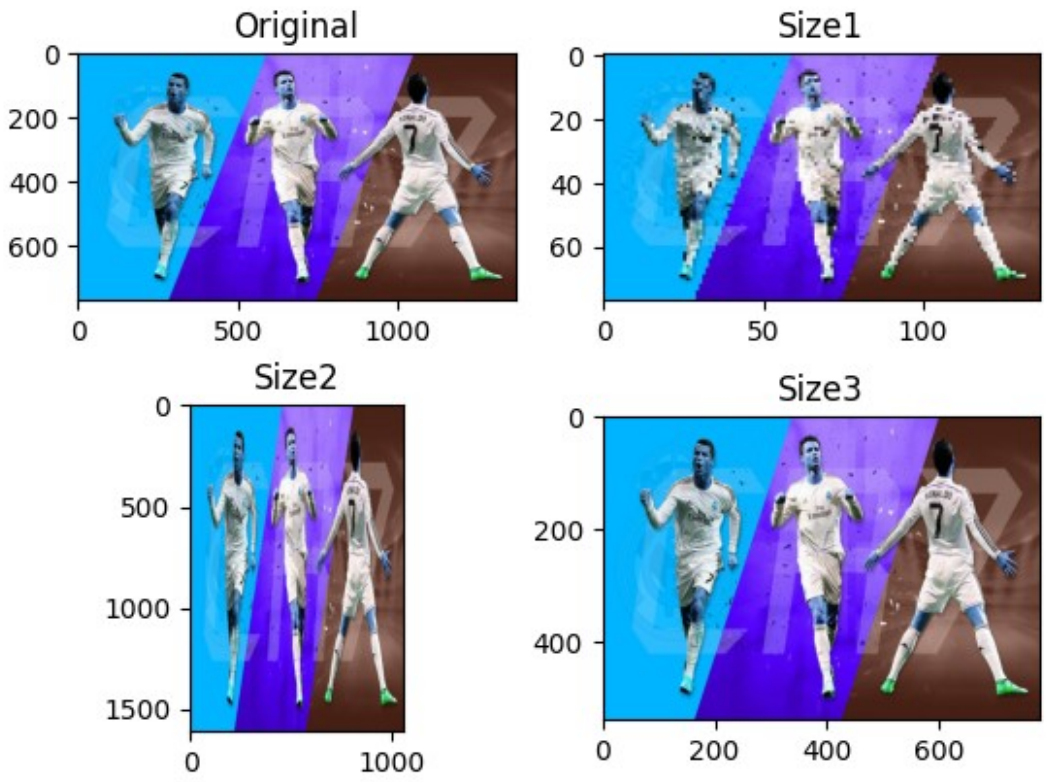
# CV

```python
#Practical-1-Program for reading,writing and displaying images

import cv2
from matplotlib import pyplot as plt
image = cv2.imread('cr7.jpg')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.imshow(image_rgb)
plt.axis('off')
plt.show()
output_path = 'image1.jpg'
cv2.imwrite(output_path, image)
print(f"Image written to {output_path}")
```


```
Image written to image1.jpg
```

```python
#Practical-2-Program for changing color spaces

import cv2
from google.colab.patches import cv2_imshow

# Load your image
path = 'cr7.jpg'  # Replace with the path to your image
src = cv2.imread(path)

# Convert the image to grayscale
image = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)

# Display the image using cv2_imshow
cv2_imshow(image)
```


```
#Practical-3-Program for resizing images

import cv2
import numpy as np
import matplotlib.pyplot as plt
image = cv2.imread(r'cr7.jpg')
# Loading the image
size1 = cv2.resize(image, (0, 0), fx = 0.1, fy = 0.1)
size2 = cv2.resize(image, (1050, 1610))
stretch_near = cv2.resize(image, (780, 540), interpolation =
cv2.INTER_LINEAR)
Titles =["Original", "Size1", "Size2", "Size3"]
images =[image, size1, size2, stretch_near]
count = 4
for i in range(count):
  plt.subplot(2, 2, i + 1)
  plt.title(Titles[i])
  plt.imshow(images[i])
plt.show()
```




```
#Practical-4-Program to rotate image

import cv2
from matplotlib import pyplot as plt

image = cv2.imread('cr7.jpg')
num_rows, num_cols = image.shape[:2]
rotation_matrix = cv2.getRotationMatrix2D((num_cols / 2, num_rows /
2), 180, 0.6)
img_rotation = cv2.warpAffine(image, rotation_matrix, (num_cols,
num_rows))
img_rotation_rgb = cv2.cvtColor(img_rotation, cv2.COLOR_BGR2RGB)
plt.imshow(img_rotation_rgb)
plt.axis('off') # Hide axis plt.show()
```

```
(-0.5, 1365.5, 767.5, -0.5)
```

```
#Practical-5-Programs using Histogram Equalization

import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('cr7.jpg', cv.IMREAD_GRAYSCALE)
equ = cv.equalizeHist(img)
res = np.hstack((img, equ))
cv.imwrite('res.png', res)
plt.figure(figsize=(10, 5))
plt.imshow(res, cmap='gray')
plt.axis('off')
plt.show()
print('Image written to res.png')
```

```
Image written to res.png

import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('cr7.jpg', cv.IMREAD_GRAYSCALE)
```

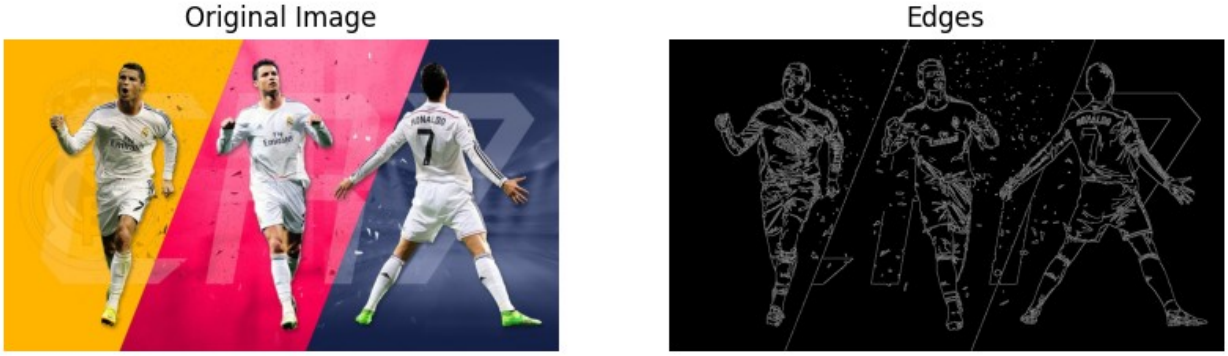
```
clahe = cv.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
cl1 = clahe.apply(img)
cv.imwrite('clahe_2.jpg', cl1)
plt.figure(figsize=(6, 6))
plt.imshow(cl1, cmap='gray')
plt.axis('off') # Hide axis plt.show()
print('Image written to clahe_2.jpg')
```

```
Image written to clahe_2.jpg
```


```
#Practical-6-Program for Edge Detection

import cv2
import matplotlib.pyplot as plt
img = cv2.imread('cr7.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2GRAY)
ret, thresh = cv2.threshold(gray, 125, 255, 0)
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
copy_img = img_rgb.copy()
cv2.drawContours(copy_img, contours, -1, (0, 0, 255), 2)
fig, axes = plt.subplots(1, 3, figsize=(20, 10))
axes[0].imshow(img_rgb)
axes[0].set_title('Original Image')
axes[0].axis('off')
axes[1].imshow(gray, cmap='gray')
axes[1].set_title('Grayscale Image')
axes[1].axis('off')
axes[2].imshow(copy_img)
axes[2].set_title('Image with Contours')
```

```
axes[2].axis('off')
plt.show()
```


```
#Practical-7-Program for Line Detection

import cv2
import matplotlib.pyplot as plt
img = cv2.imread('cr7.jpg')
edges = cv2.Canny(img, 100, 200, apertureSize=3, L2gradient=True)
plt.imsave('test.png', edges, cmap='gray', format='png')
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(1, 2, 2)
plt.title('Edges')
plt.imshow(edges, cmap='gray')
plt.axis('off')
plt.show()
print("Edges image saved as 'test.png'")
```

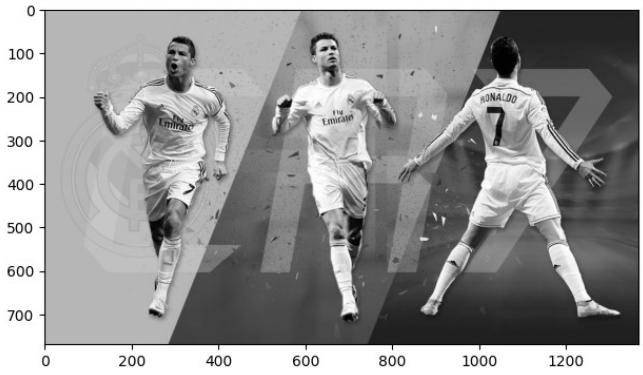


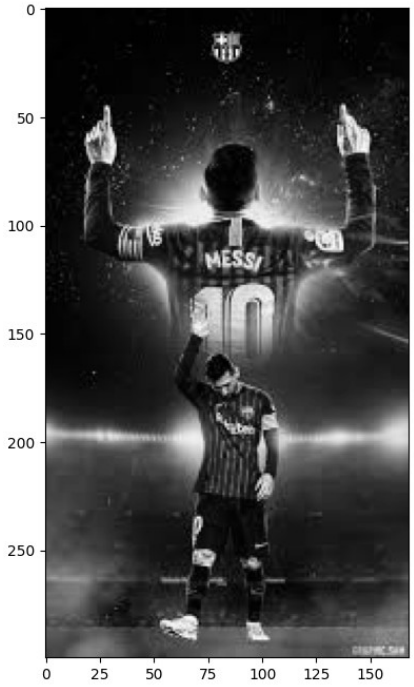

```
Edges image saved as 'test.png'

#Practical-8-Programs using Scale Invariant Feature Transform (SIFT)

import cv2
import matplotlib.pyplot as plt
%matplotlib inline
```

```
img1 = cv2.imread('cr7.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
sift = cv2.SIFT_create()
keypoints_1, descriptors_1 = sift.detectAndCompute(gray1, None)
img_1 = cv2.drawKeypoints(img1, keypoints_1, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
plt.figure(figsize=(10, 10))
plt.imshow(cv2.cvtColor(img_1, cv2.COLOR_BGR2RGB))
plt.title('Image with SIFT Keypoints')
plt.axis('off')
plt.show()
```

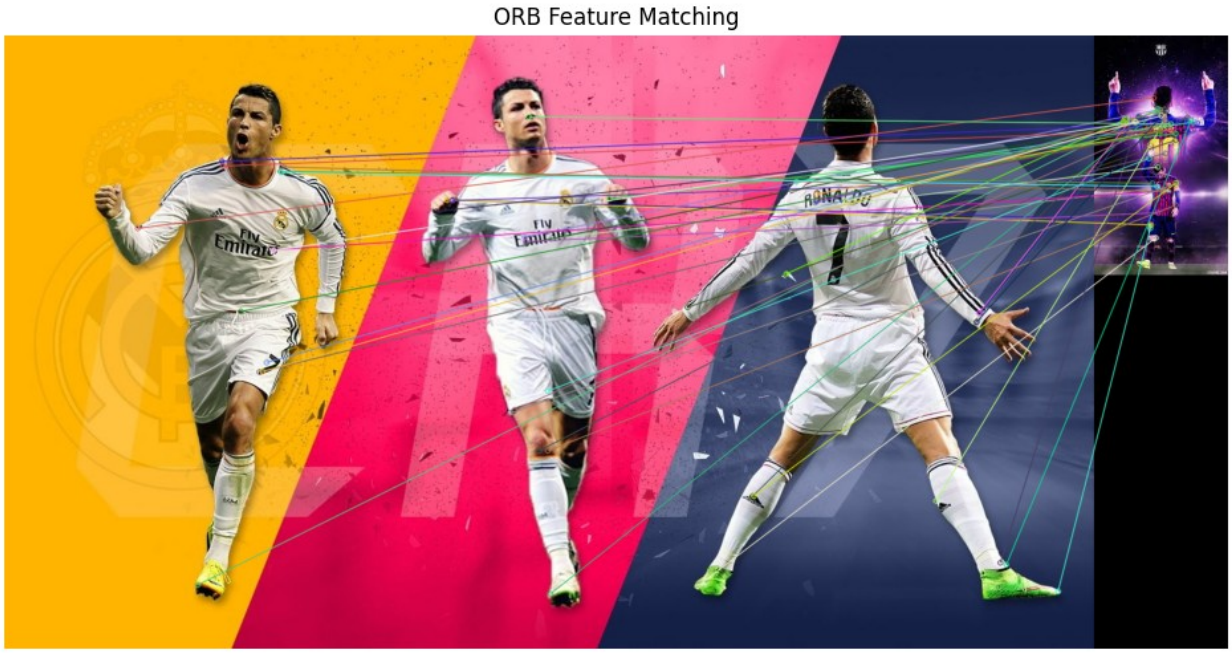

```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
img1 = cv2.imread('cr7.jpg')
img2 = cv2.imread('lm10.jpg')
img1 = cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
img2 = cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)
figure,ax = plt.subplots(1,2,figsize=(16,8))
ax[0].imshow(img1,cmap='gray')
ax[1].imshow(img2,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7e1dc04da980>
```




```
import cv2
import matplotlib.pyplot as plt
img1 = cv2.imread('cr7.jpg')
img2 = cv2.imread('lm10.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
orb = cv2.ORB_create()
keypoints_1, descriptors_1 = orb.detectAndCompute(gray1, None)
keypoints_2, descriptors_2 = orb.detectAndCompute(gray2, None)
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(descriptors_1, descriptors_2)
matches = sorted(matches, key=lambda x: x.distance)
img_matches = cv2.drawMatches(img1, keypoints_1, img2, keypoints_2,
matches[:50], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
plt.figure(figsize=(12, 6))
plt.imshow(cv2.cvtColor(img_matches, cv2.COLOR_BGR2RGB))
plt.title('ORB Feature Matching')
plt.axis('off')
plt.show()
```




```
import cv2
import matplotlib.pyplot as plt
img1 = cv2.imread('cr7.jpg')
img2 = cv2.imread('lm10.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
sift = cv2.SIFT_create()
keypoints_1, descriptors_1 = sift.detectAndCompute(gray1, None)
keypoints_2, descriptors_2 = sift.detectAndCompute(gray2, None)

bf = cv2.BFMatcher()
matches = bf.knnMatch(descriptors_1, descriptors_2, k=2)


good_matches = []
for m, n in matches:
  if m.distance < 0.75 * n.distance: good_matches.append(m)

img_matches = cv2.drawMatches(img1, keypoints_1, img2, keypoints_2,
good_matches, None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
axes[0].imshow(cv2.cvtColor(img1, cv2.COLOR_BGR2RGB))
axes[0].set_title('Image 1')
axes[0].axis('off')
axes[1].imshow(cv2.cvtColor(img_matches, cv2.COLOR_BGR2RGB))
axes[1].set_title('Matches')
axes[1].axis('off')
axes[2].imshow(cv2.cvtColor(img2, cv2.COLOR_BGR2RGB))
axes[2].set_title('Image 2')
```

```
axes[2].axis('off')
plt.tight_layout()
plt.show()
```


```
def gaussian_blur(image, kernel_size=(5, 5), sigma=0): return
cv2.GaussianBlur(image, kernel_size, sigma)
image = cv2.imread('cr7.jpg')
blurred_image = gaussian_blur(image, kernel_size=(11, 11), sigma=0)
plt.imshow(blurred_image)
```

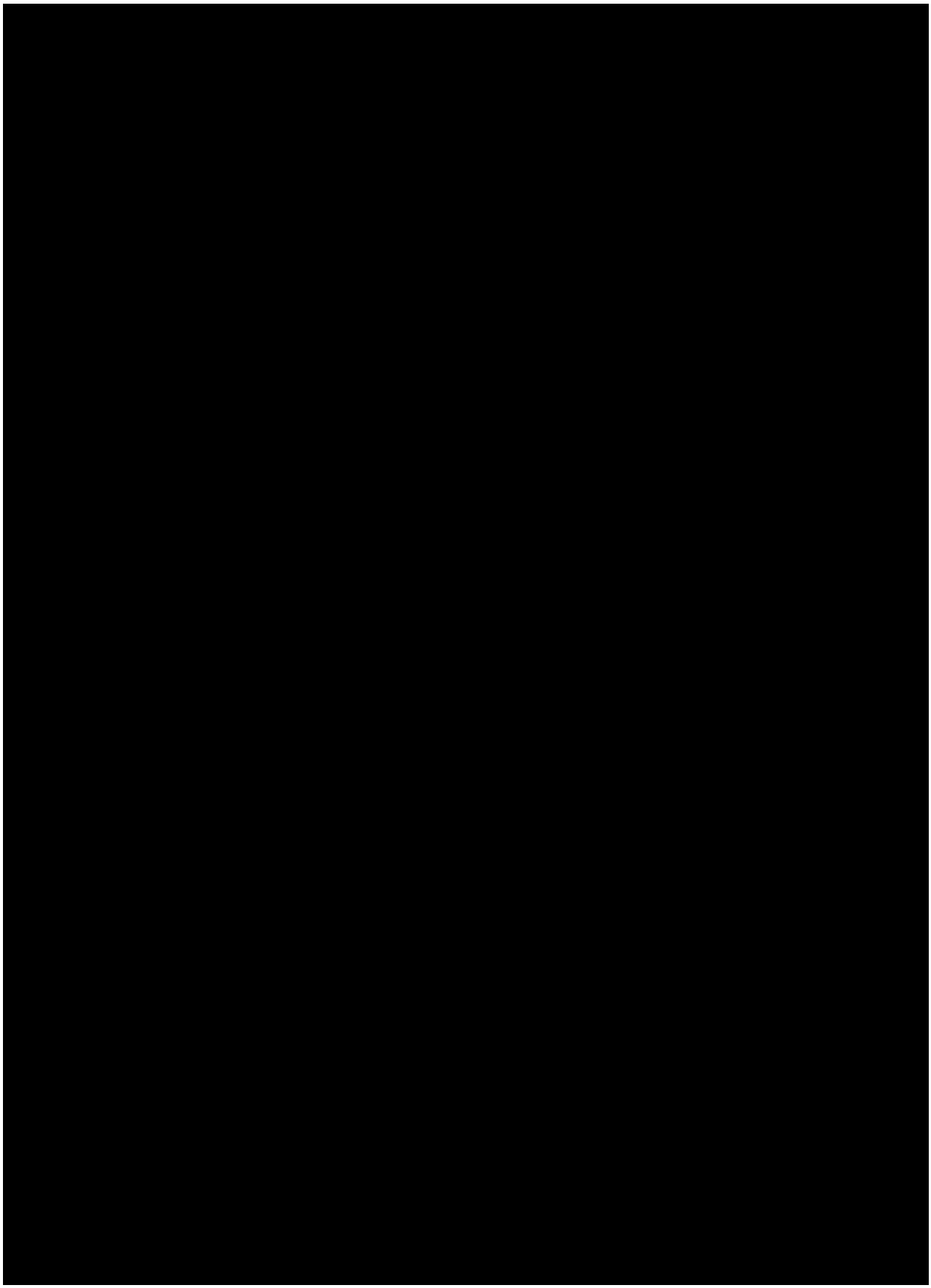
```
<matplotlib.image.AxesImage at 0x7e1dc05bfb20>
```


```
#Practical-9-Programs for Motion Detection
```

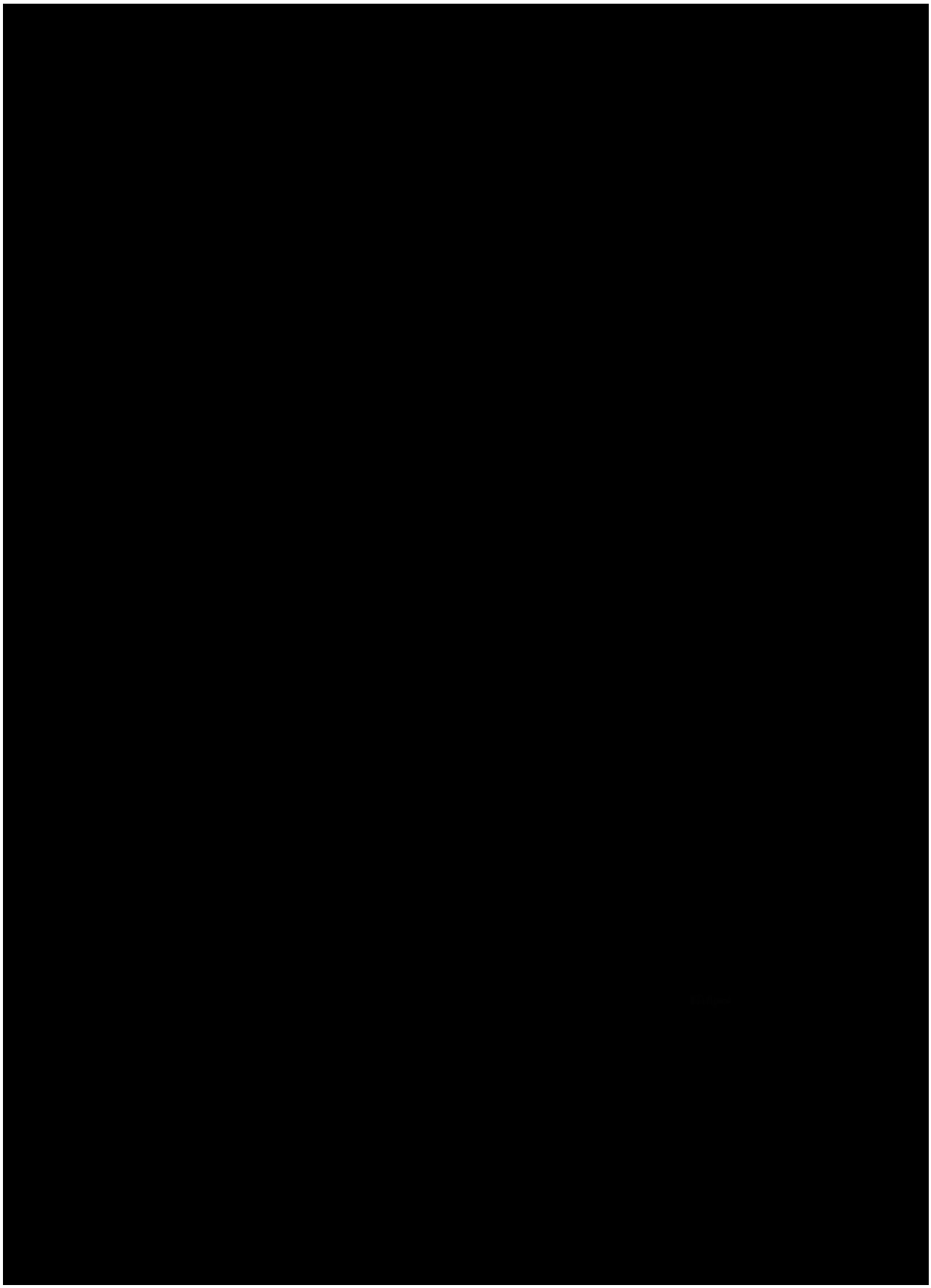
```python
from google.colab import files
import cv2
import numpy as np

# Upload video file
uploaded = files.upload()

# Assuming the uploaded file is named 'video.mp4'
cap = cv2.VideoCapture('Siiuuu.mp4')

last_mean = 0

while cap.isOpened():
    ret, frame = cap.read()

    if not ret:
        print("Failed to grab frame or end of video")
        break

    # Display the resulting frame
    # In Colab, you would typically save and display the frame using cv2_imshow
    from google.colab.patches import cv2_imshow
    cv2_imshow(frame)

    # Convert the frame to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Calculate the mean difference
    result = np.abs(np.mean(gray) - last_mean)
    print(result)

    # Detect change
    if result > 0:
        print("Change detected")
    else:
        print("Change not detected")

    # Update last_mean
    last_mean = np.mean(gray)

    # Wait for a while before showing the next frame
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything is done, release the capture
cap.release()
cv2.destroyAllWindows()
```

```
<IPython.core.display.HTML object>
```

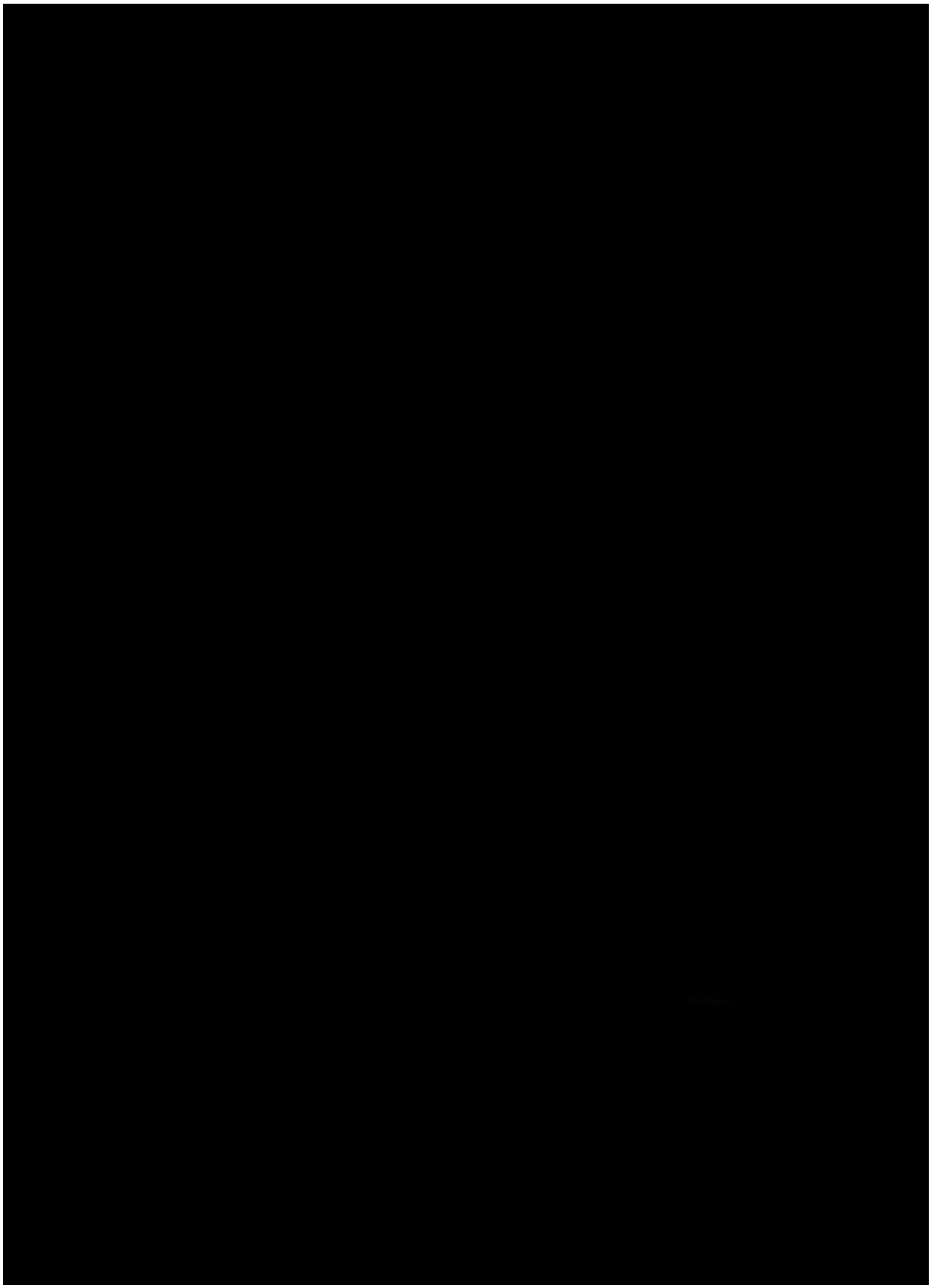
```
Saving Siiuuu.mp4 to Siiuuu (1).mp4
```



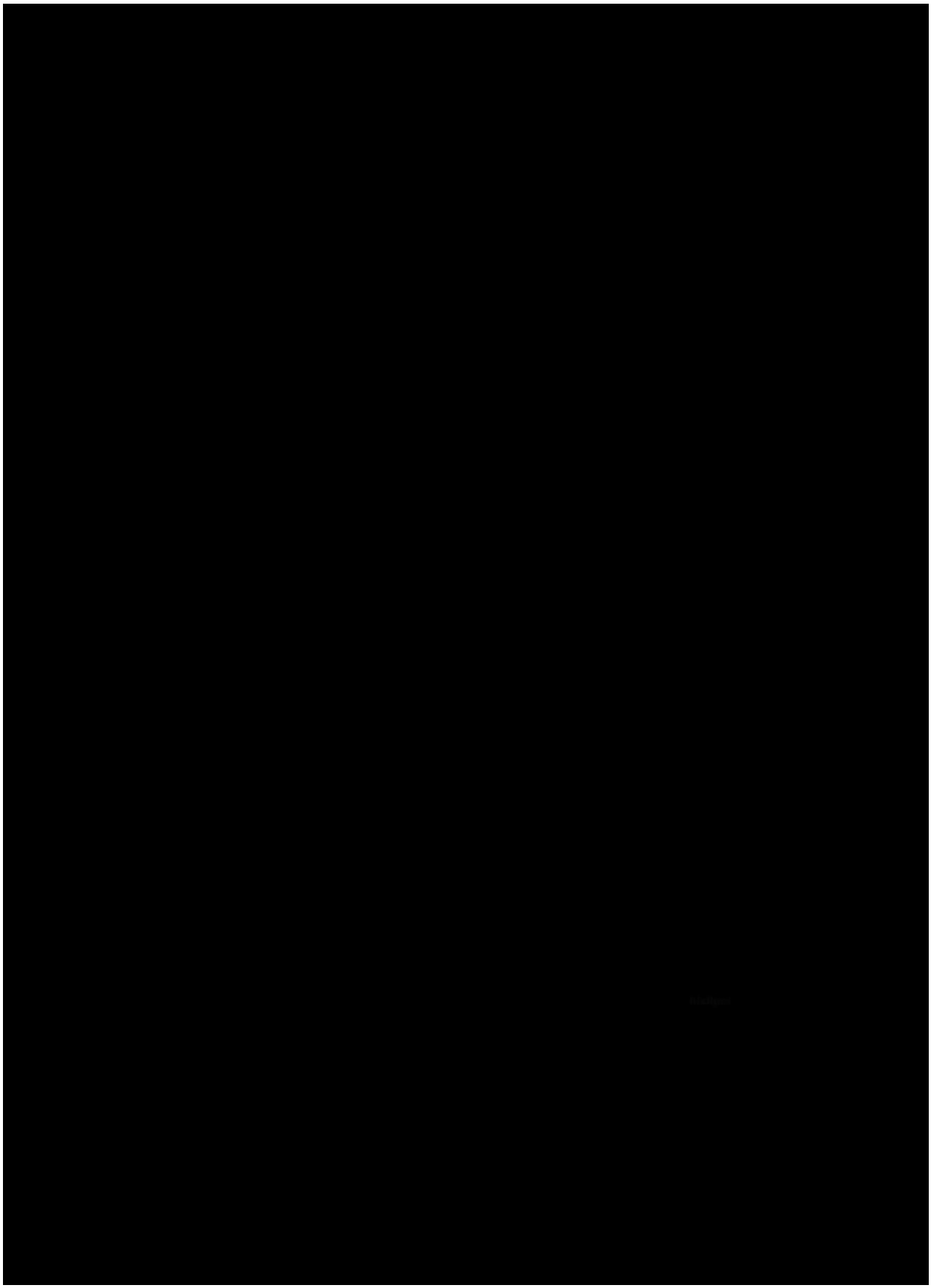
```
0.0
Change not detected
```



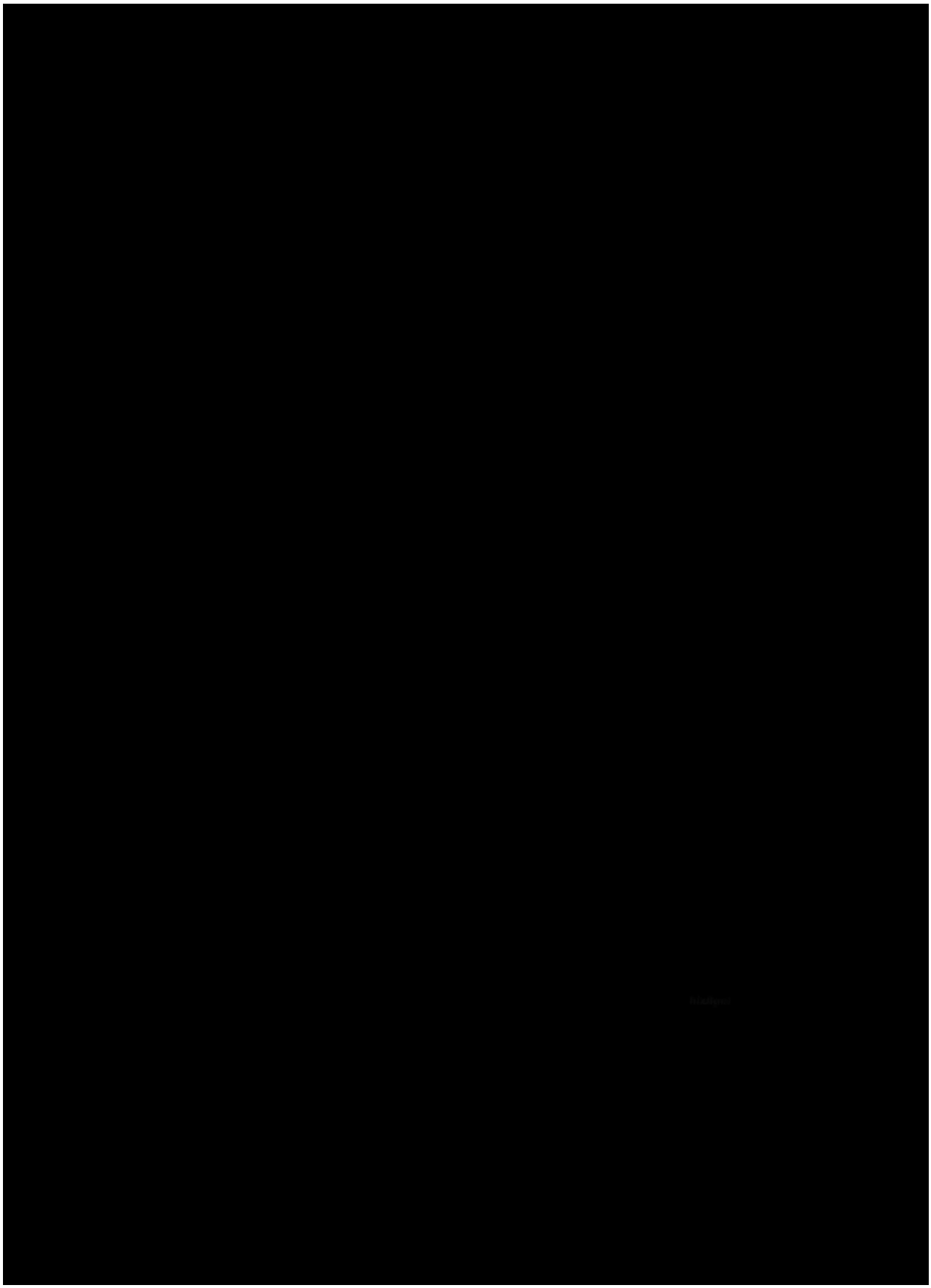
```
0.00038773148148148147
Change detected
```



```
0.00024353780864197532
Change detected
```

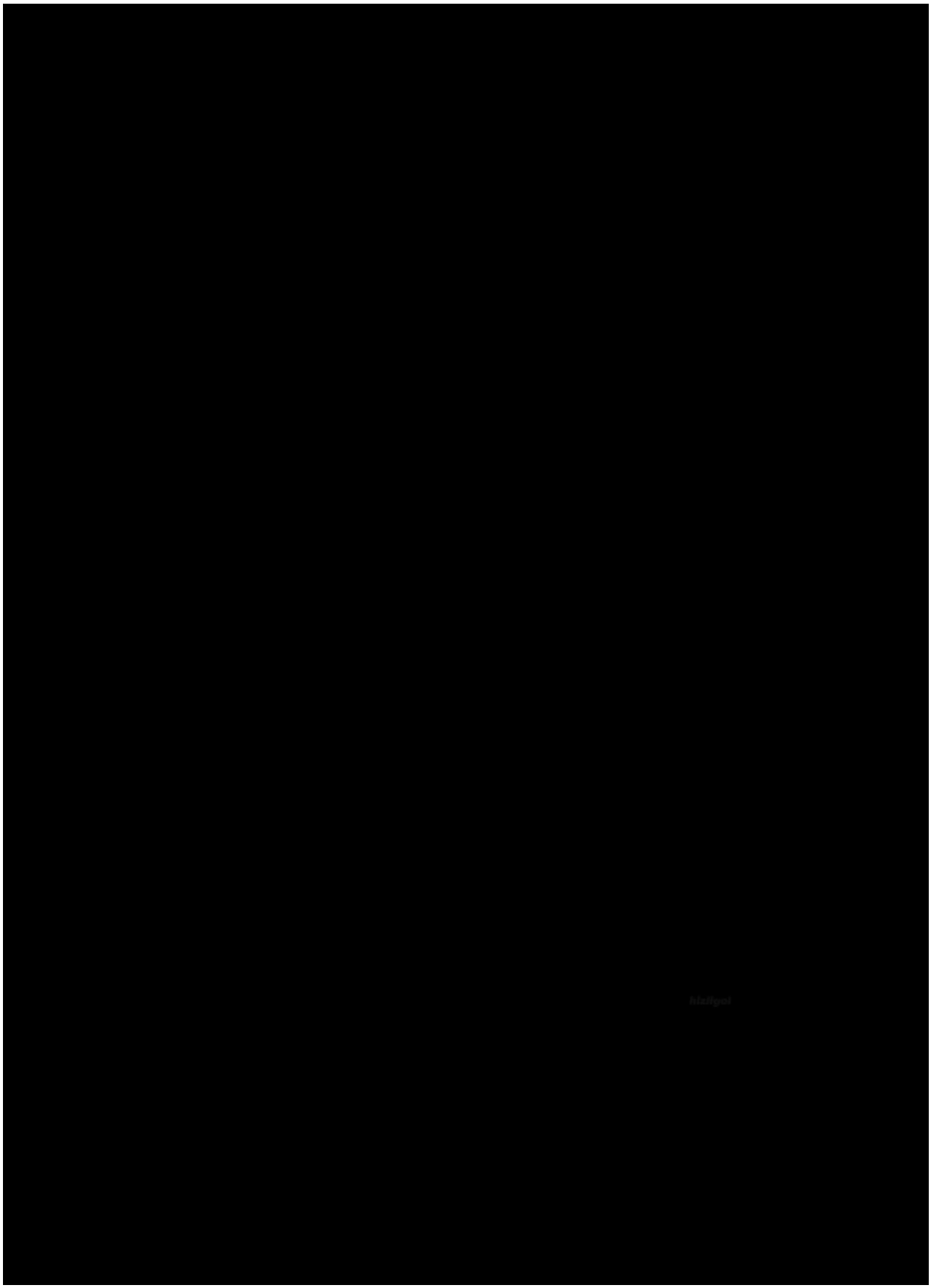


```
0.00030140817901234567
Change detected
```



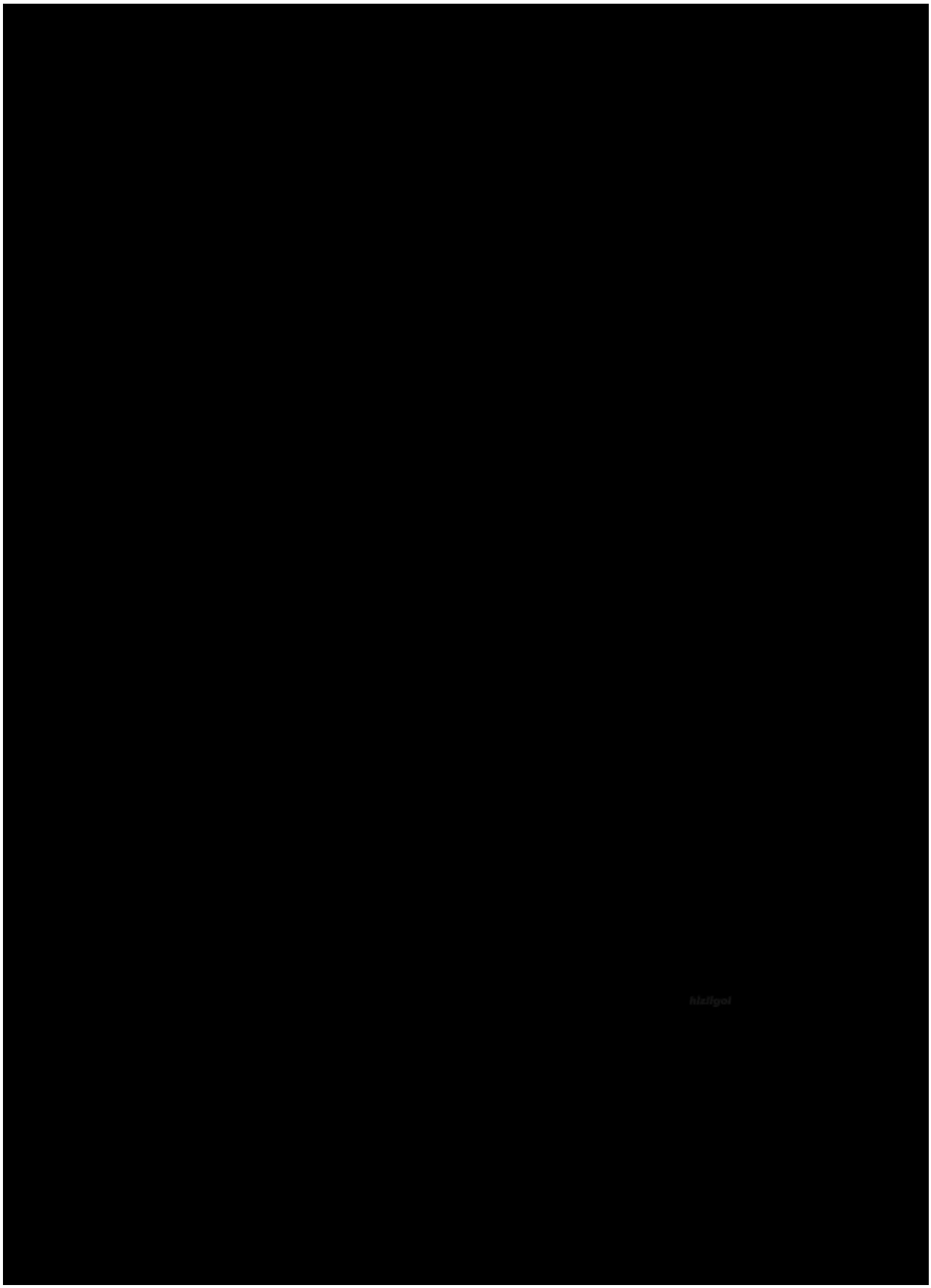
```
0.00022665895061728392
Change detected
```

```
0.00073254243827160 49
Change detected
```



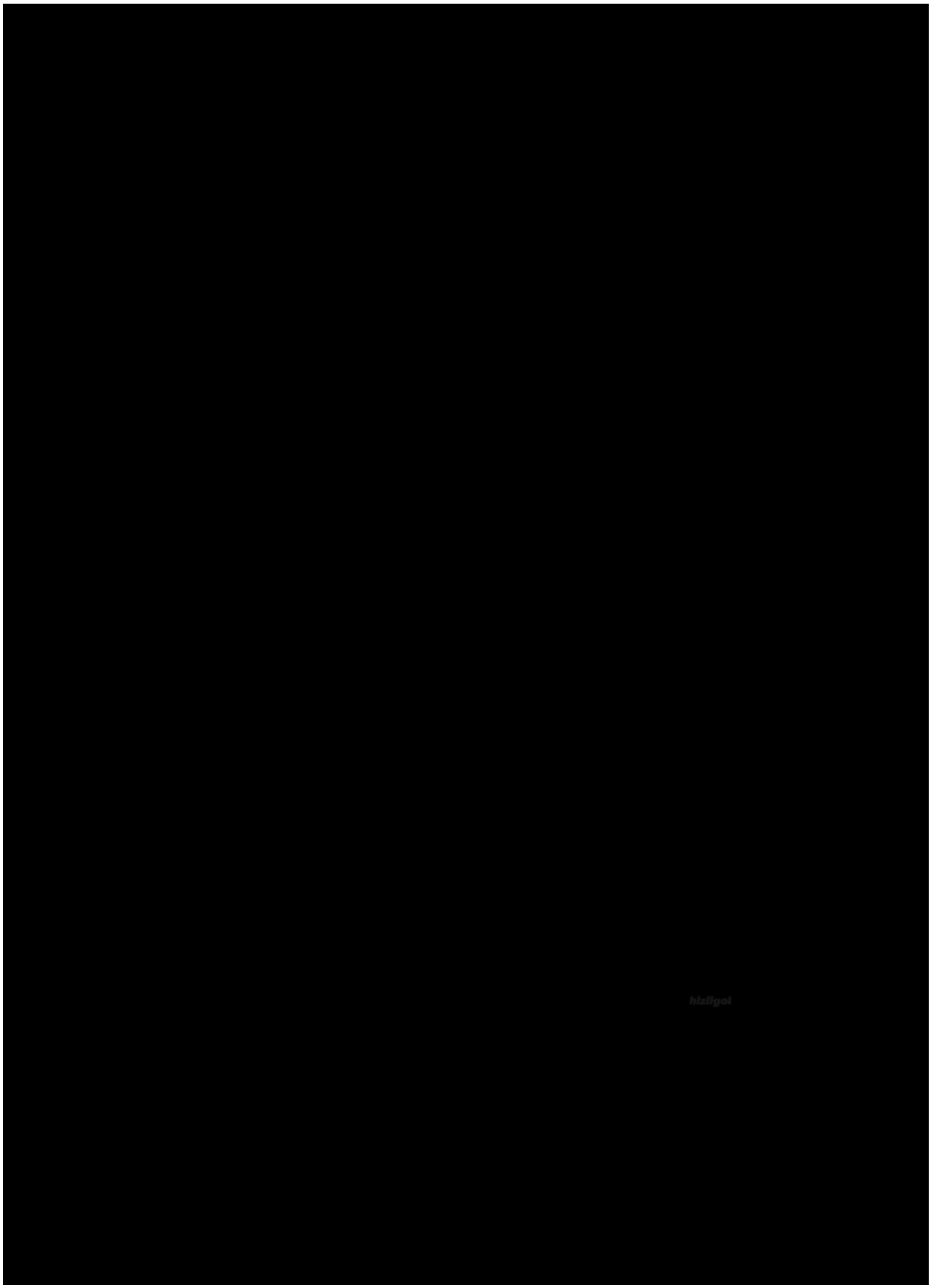
```
0.00031491126543209 88
Change detected
```

```
0.0005135995370370372
Change detected
```

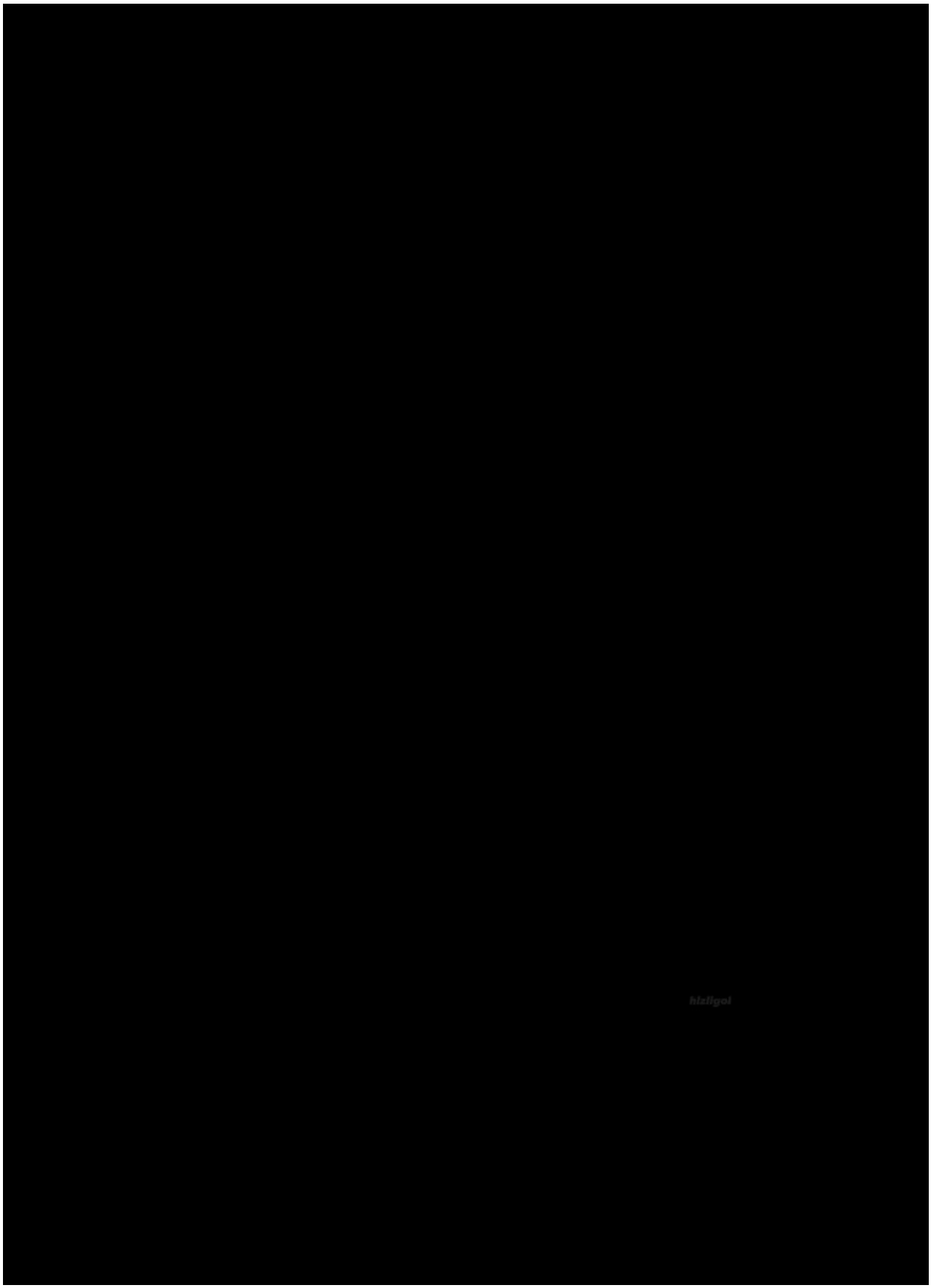


```
0.0003583140432098764
Change detected
```

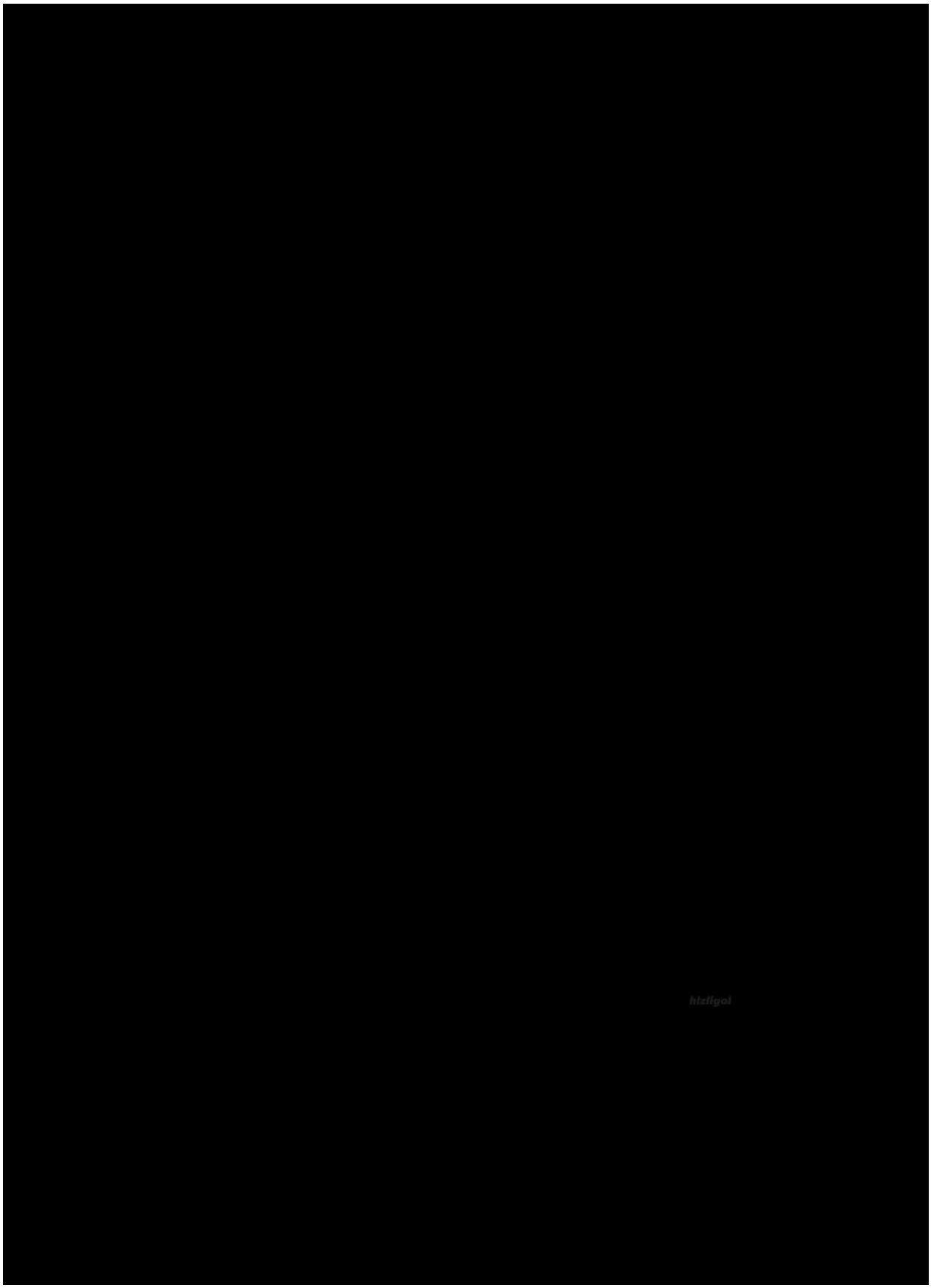
```
0.00039930555555555555
Change detected
```



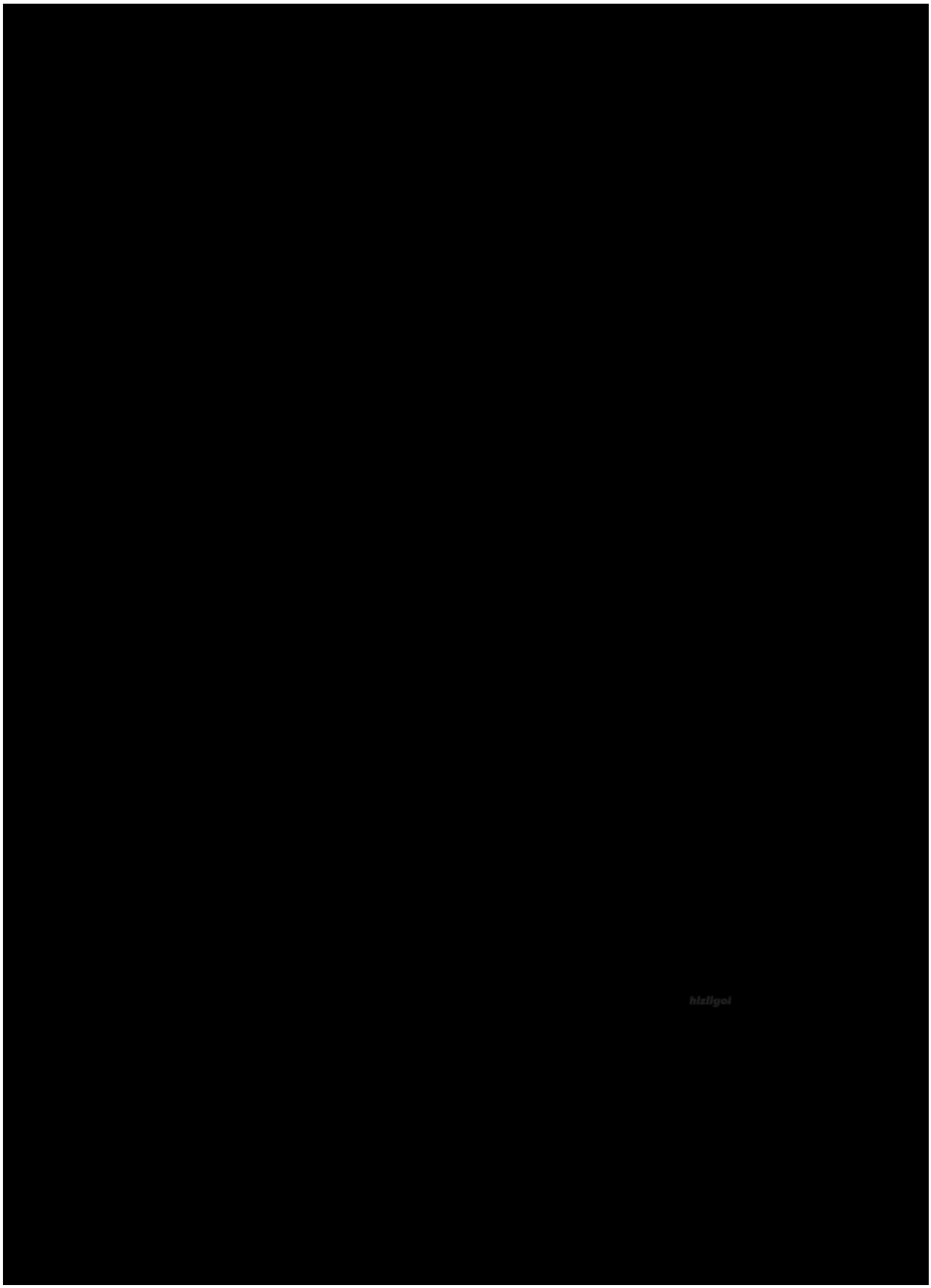
```
0.00026620370370370383
Change detected
```



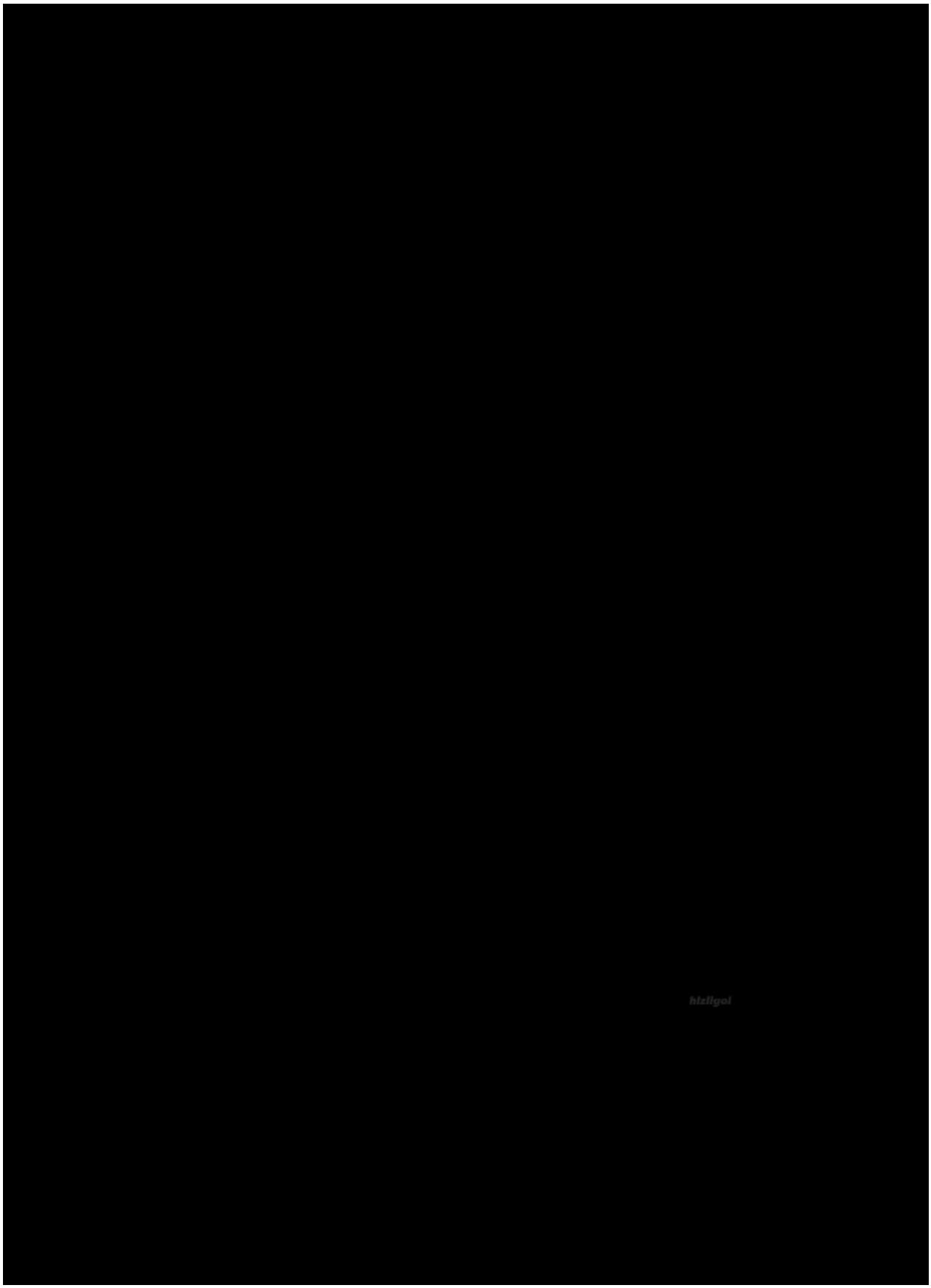
```
0.00035927854938271574
Change detected
```



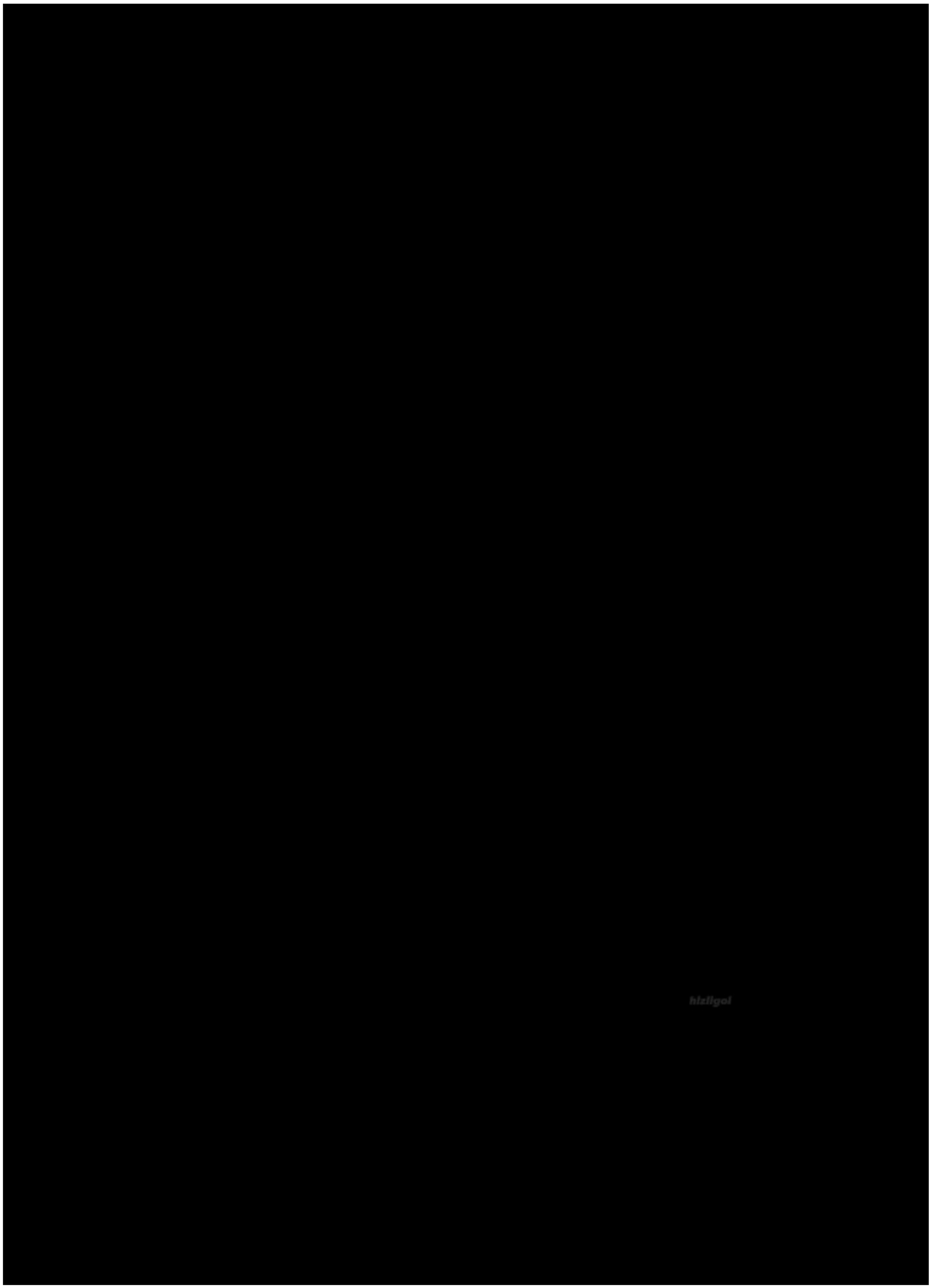
```
0.00018904320987654308
Change detected
```


hizligal

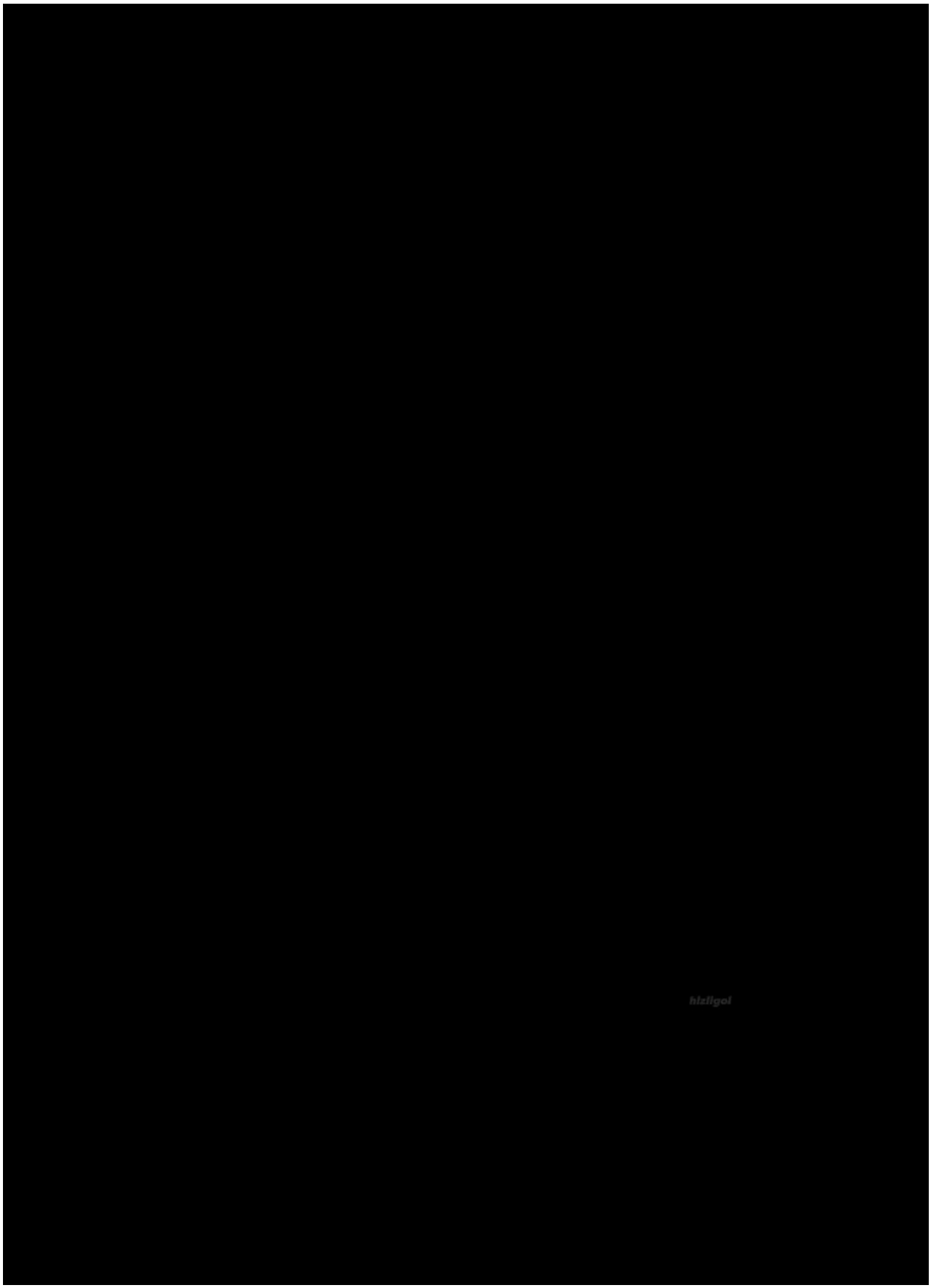
```
0.0005671296296296301
Change detected
```


hizligoi

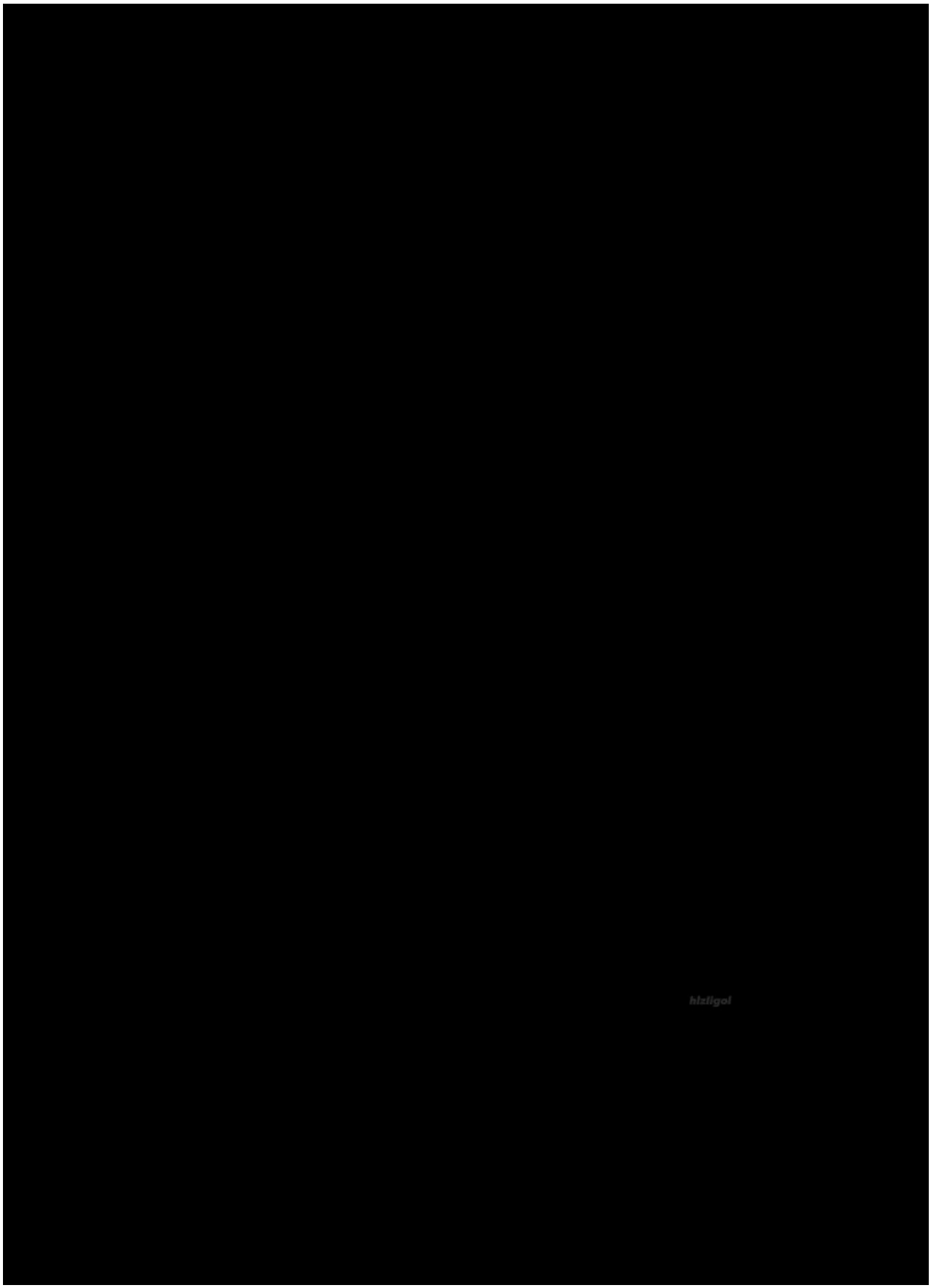
```
0.0003896604938271606
Change detected
```


hizligal

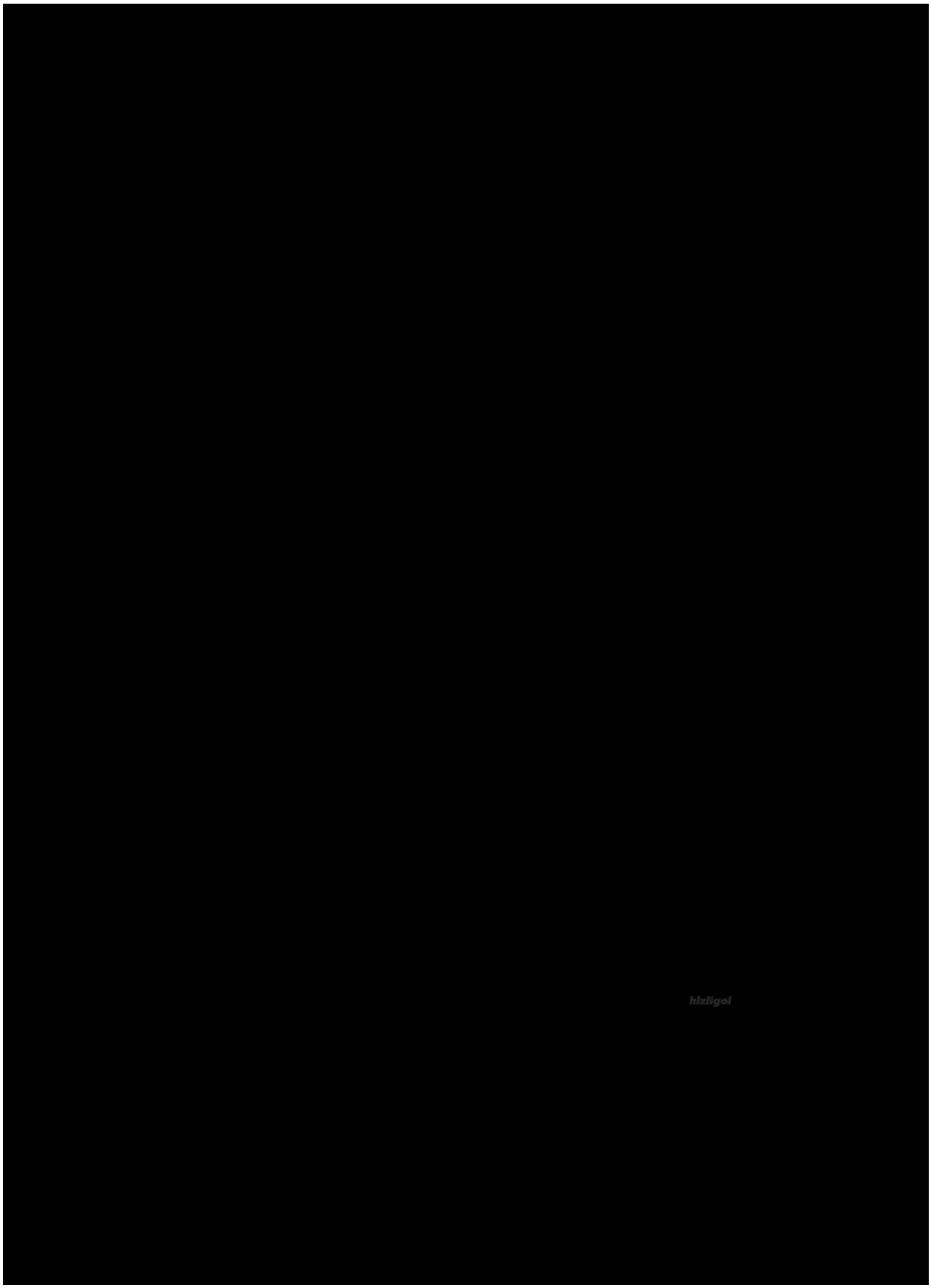
```
0.00012827932098765423
Change detected
```


hizligol

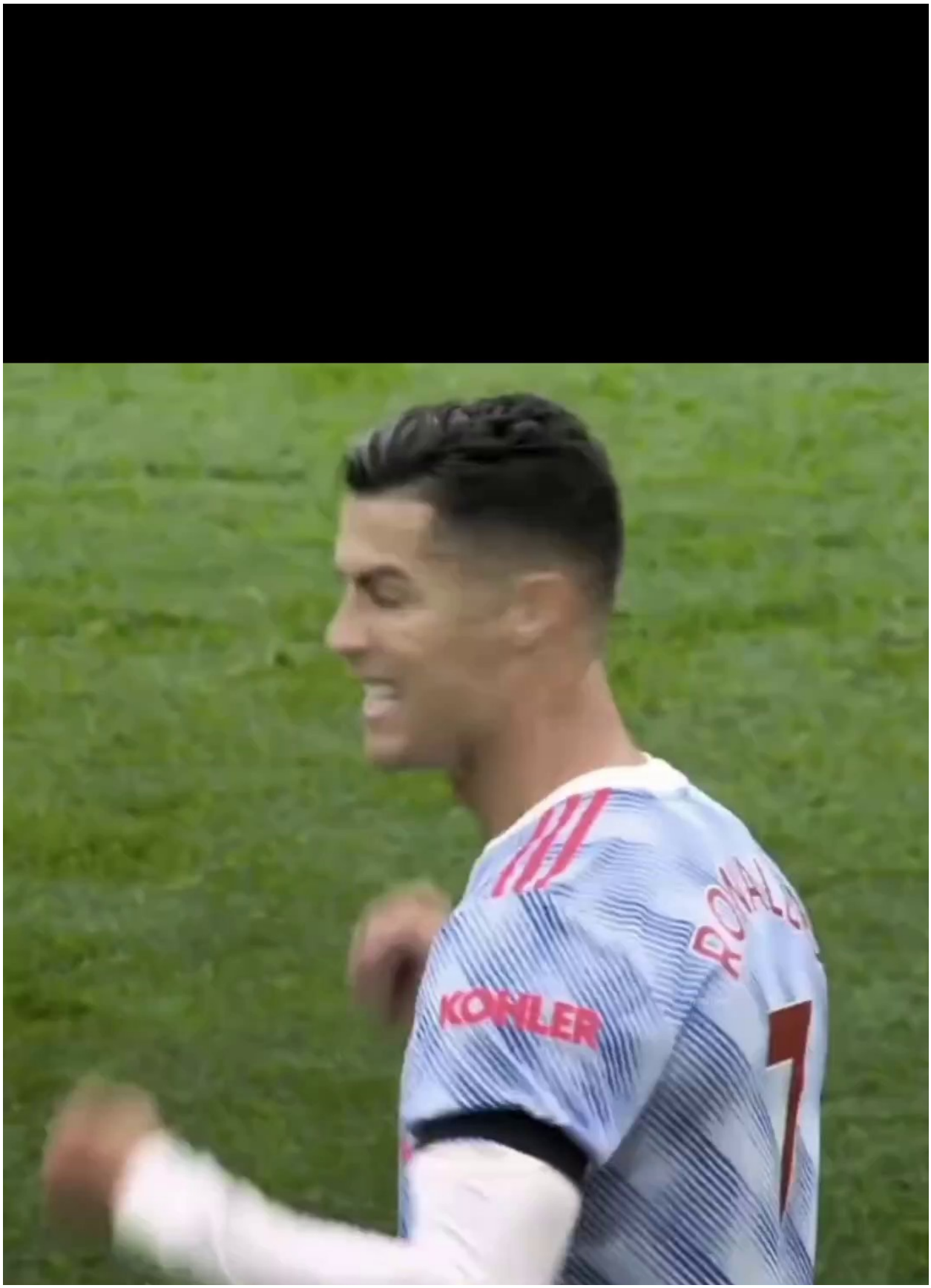
```
7.908950617283896e-05
Change detected
```


hizligal

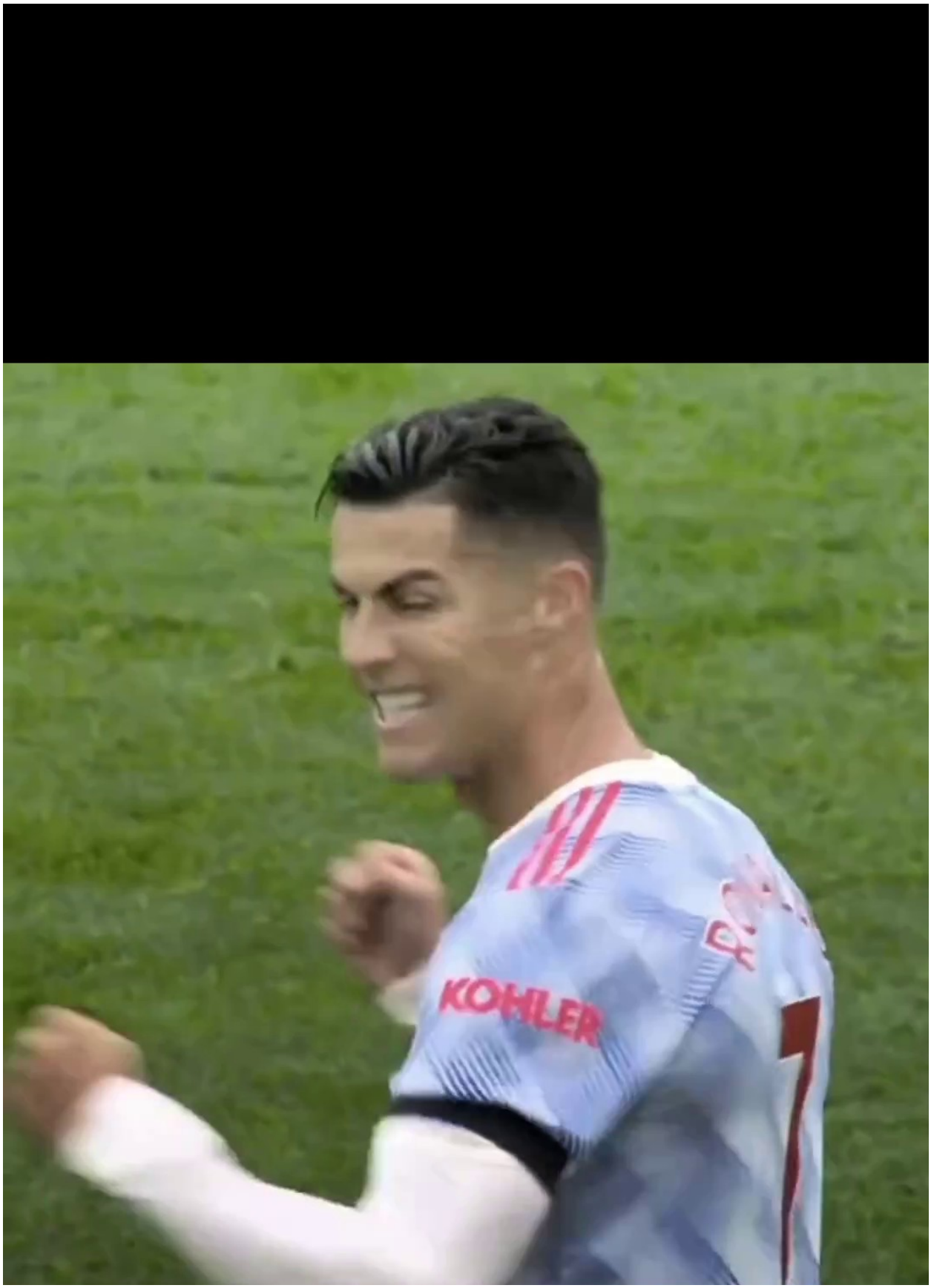
```
0.00059027777777777781
Change detected
```


hizligal

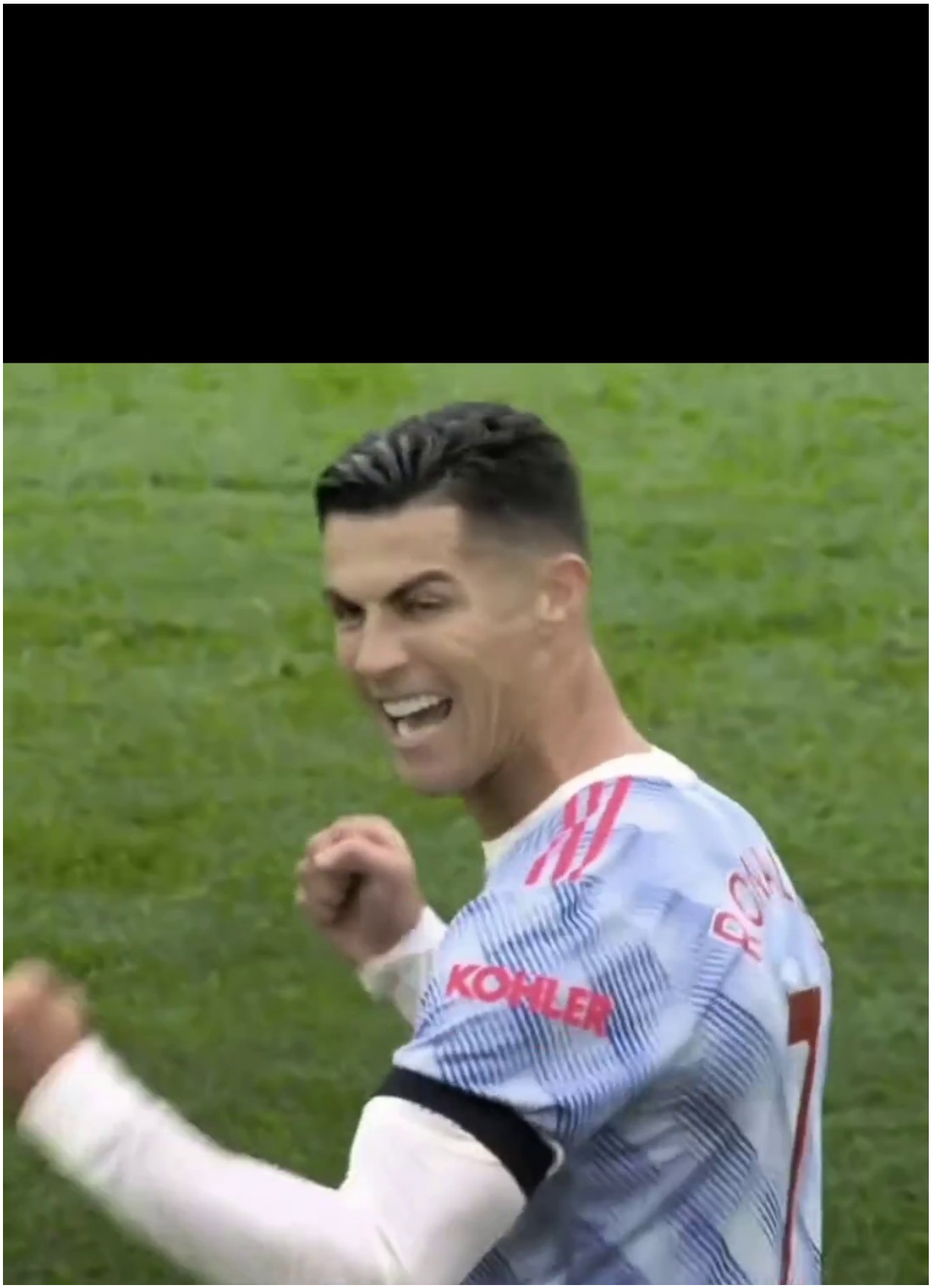
```
6.269290123456835e-05
Change detected
```


KOHLER

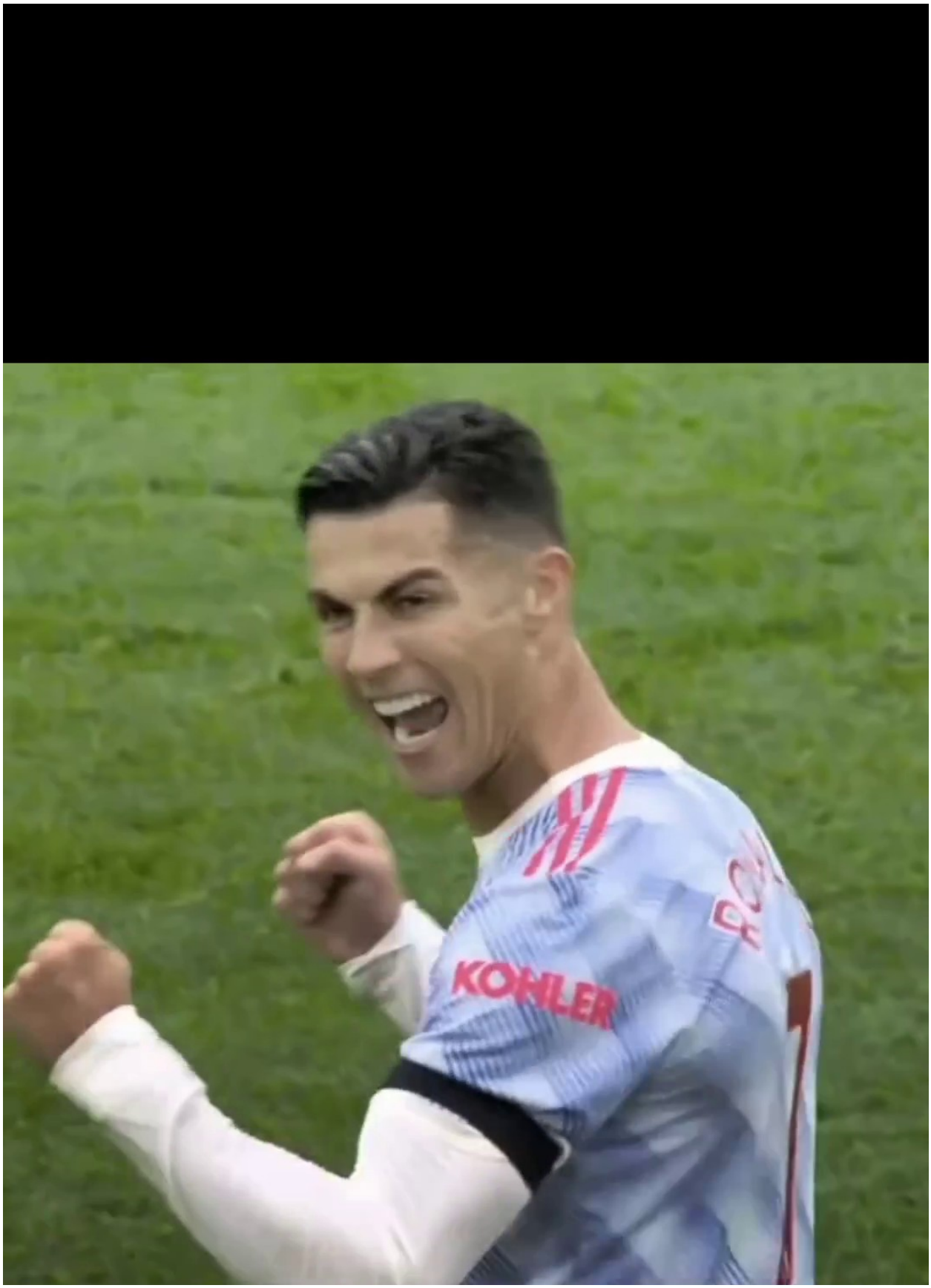
```
68.73231047453703
Change detected
```


KOHLER

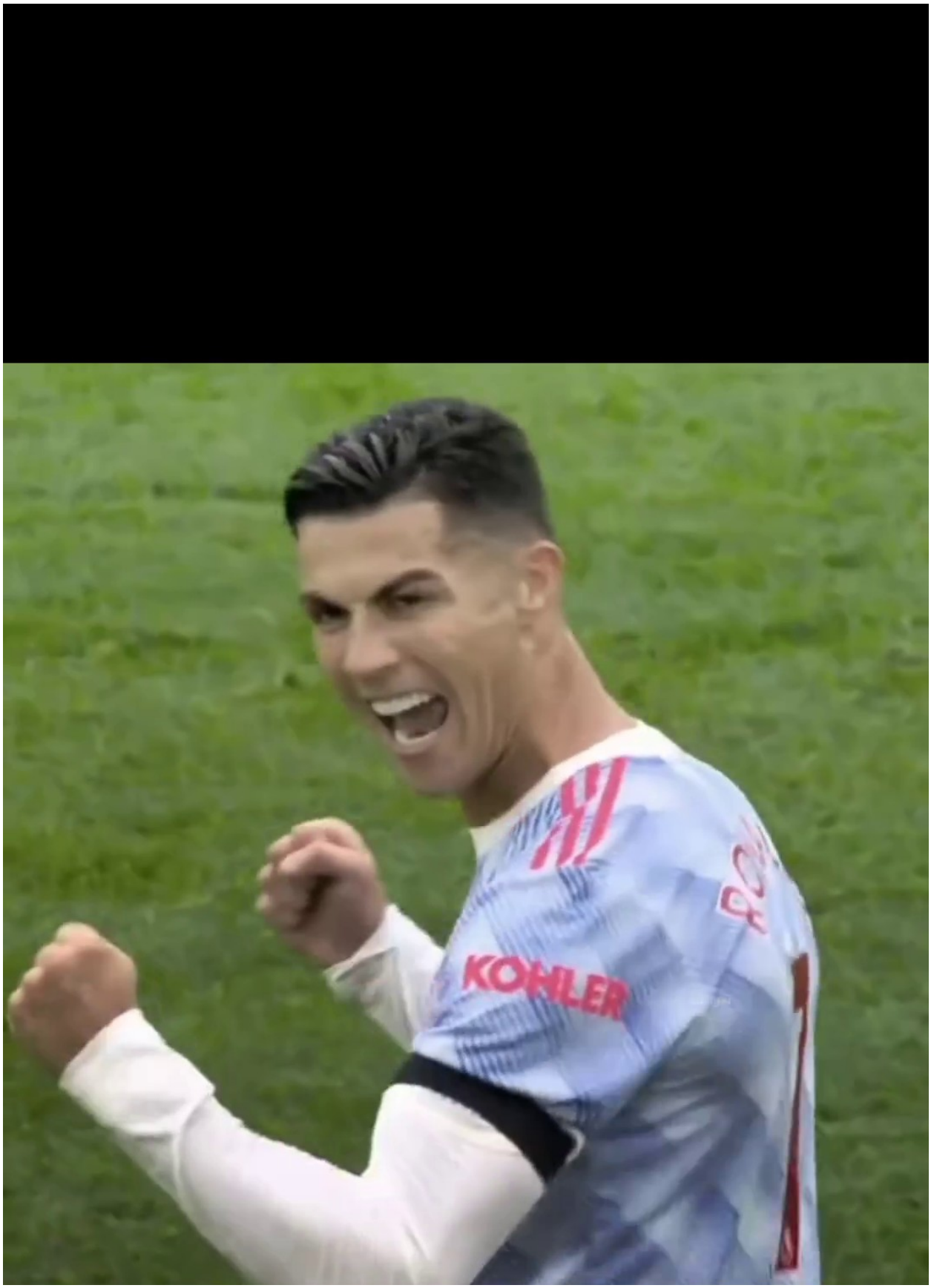
```
1.6368822337962996
Change detected
```


KOHLER

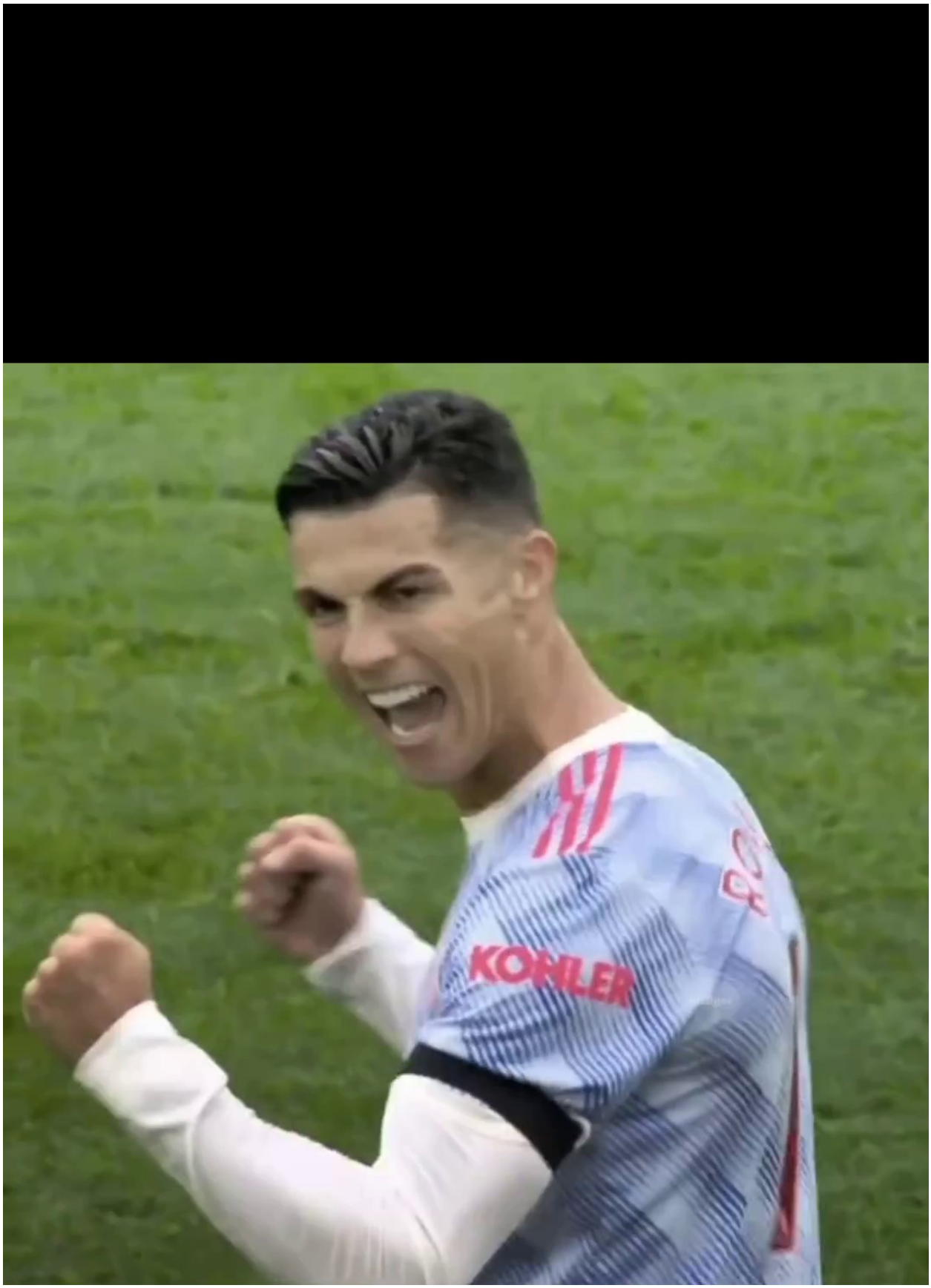
```
0.22717062114197972
Change detected
```


KOHLER

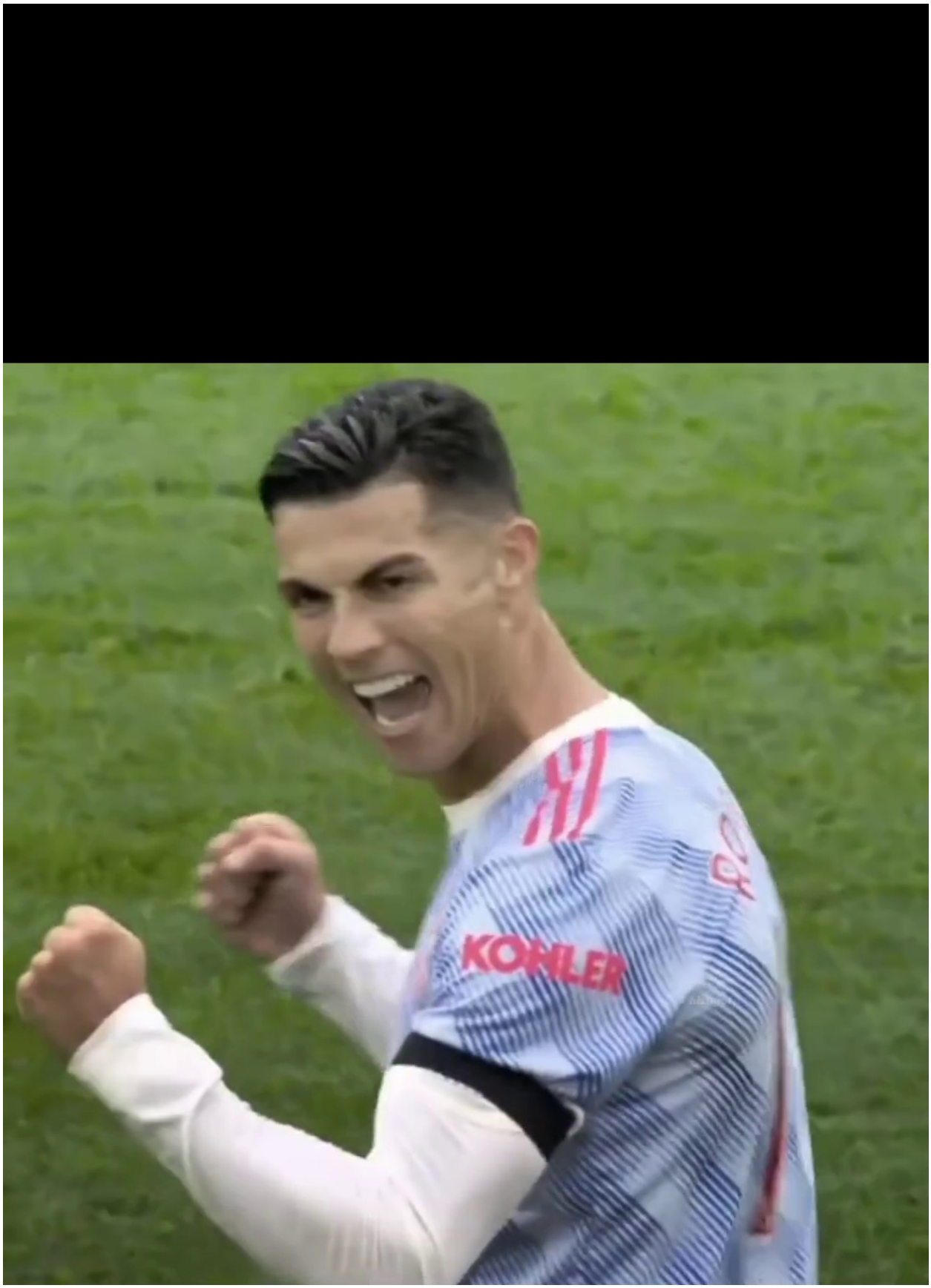
```
0.16228684413580652
Change detected
```


KOHLER

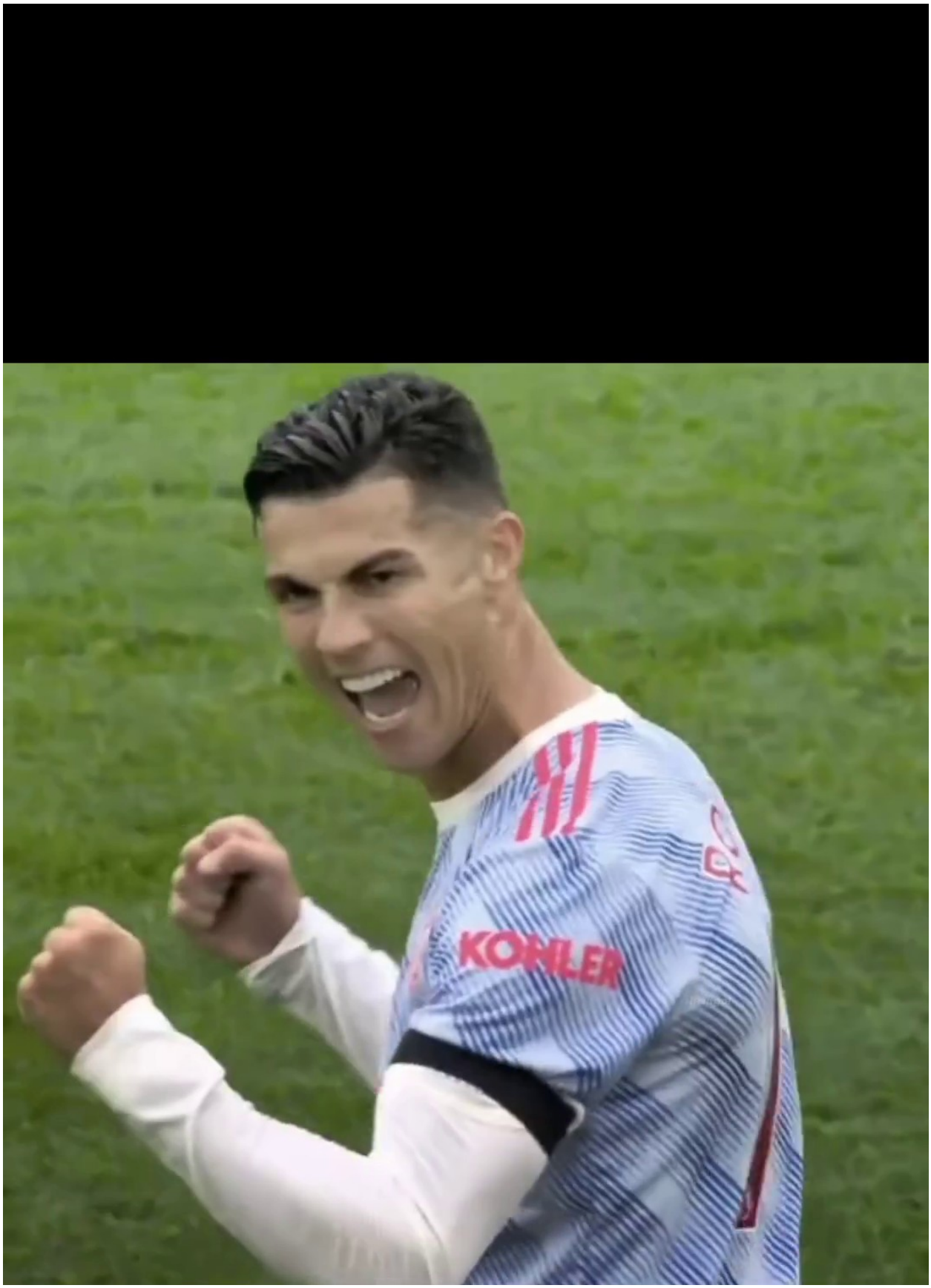
```
0.17955295138888516
Change detected
```


KOHLER

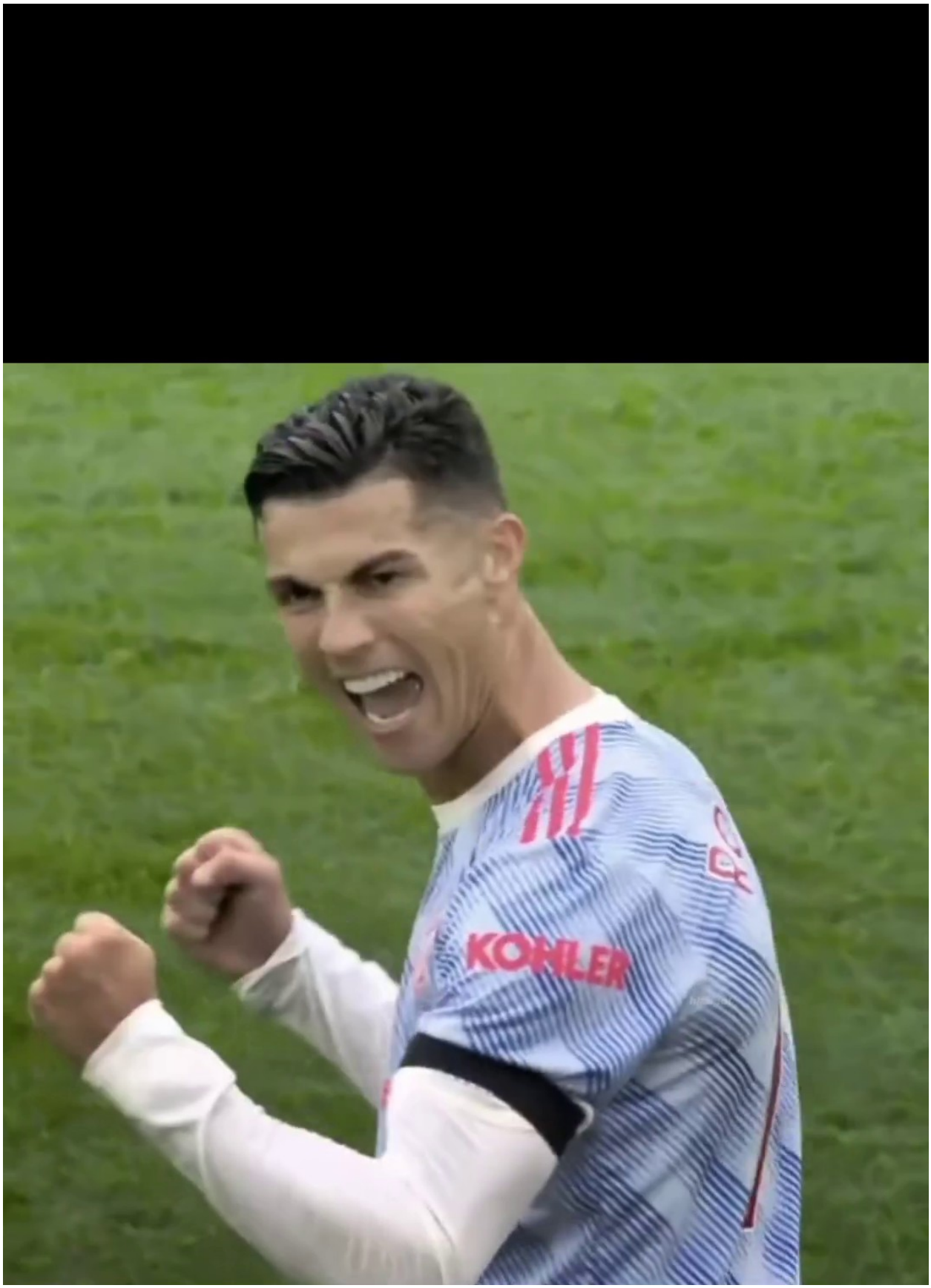
```
0.11328366126544154
Change detected
```


KOHLER

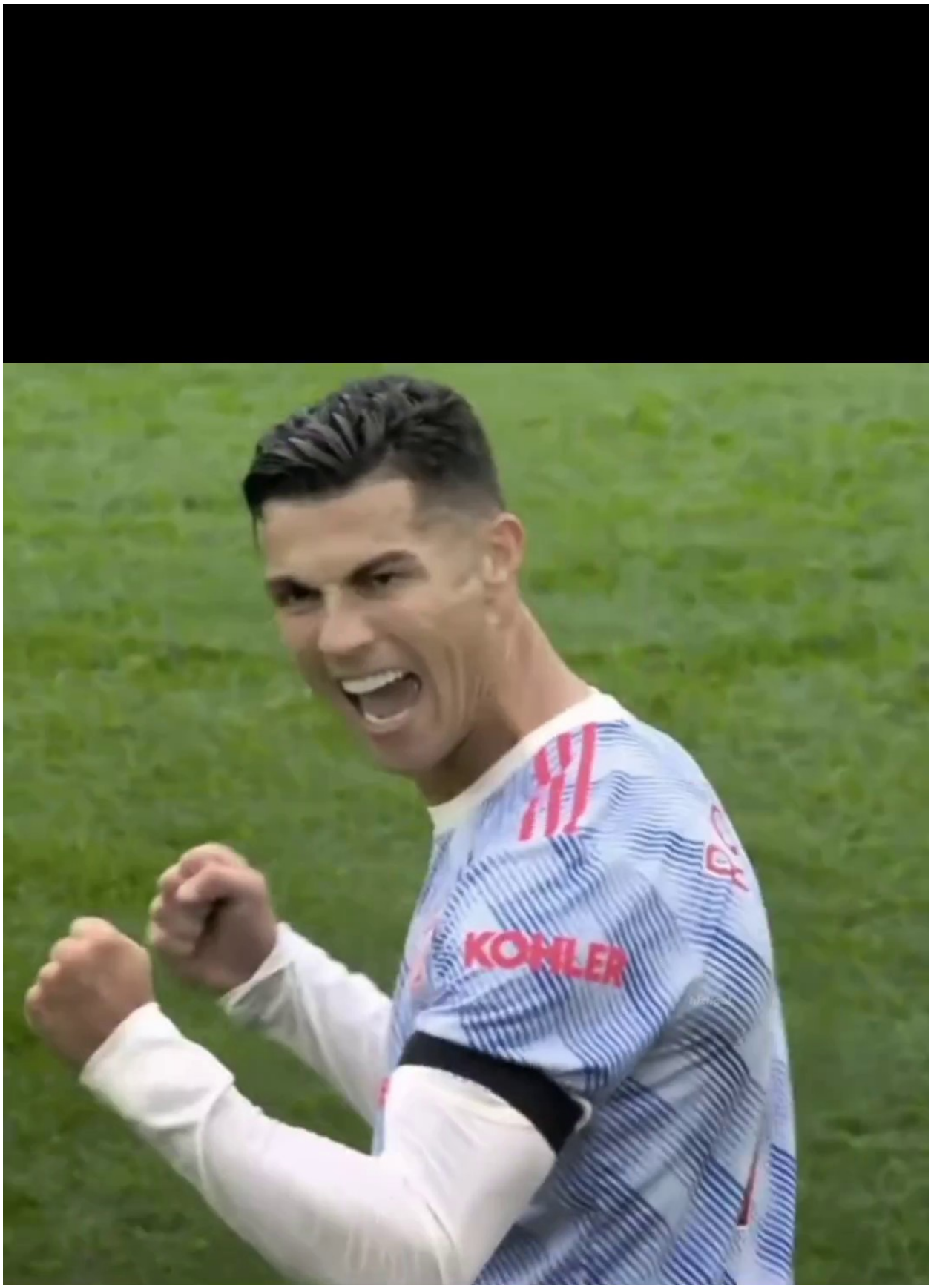
```
0.49167824074075384
Change detected
```


KOHLER

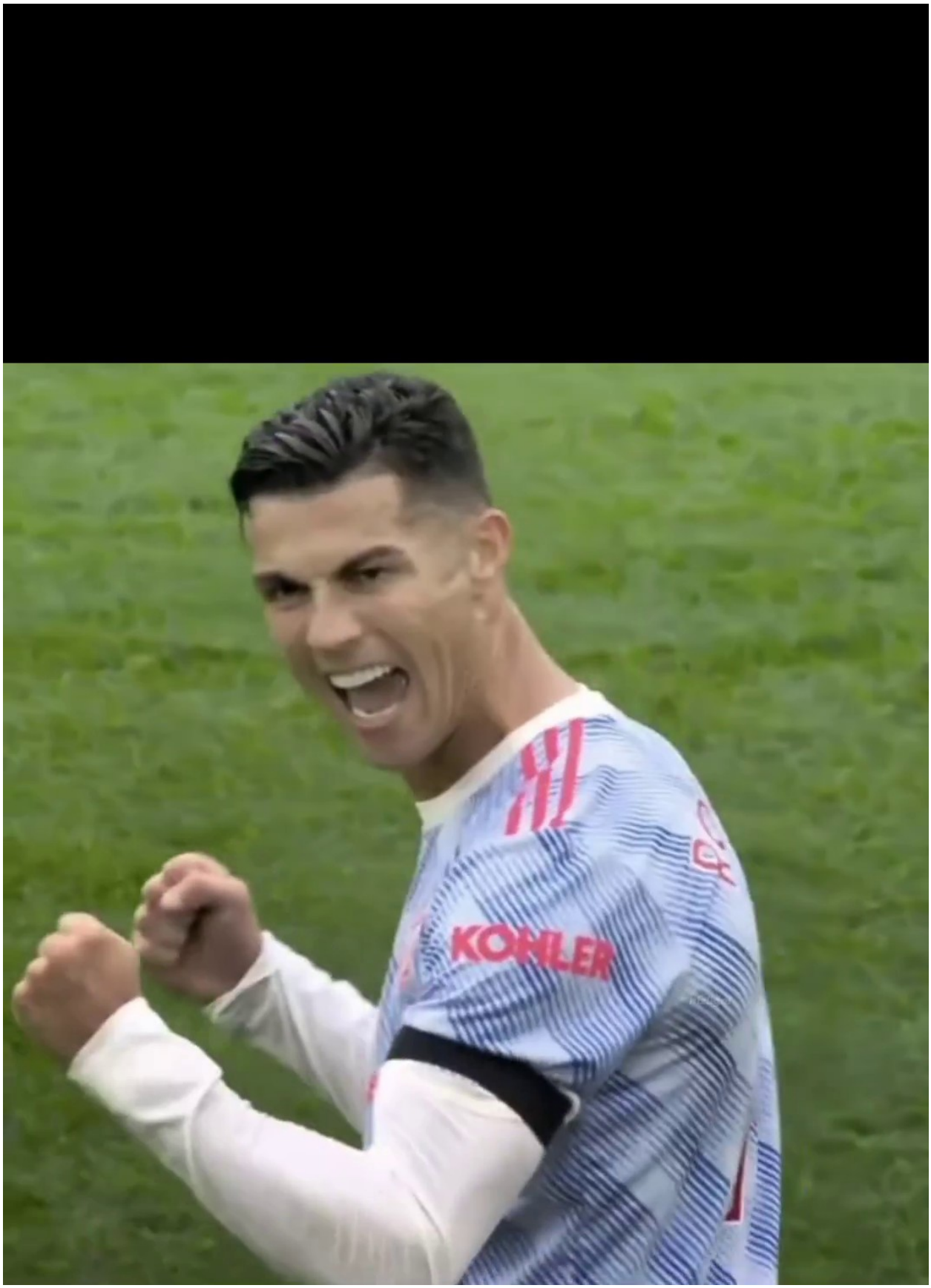
```
0.22325520833334167
Change detected
```


KOHLER

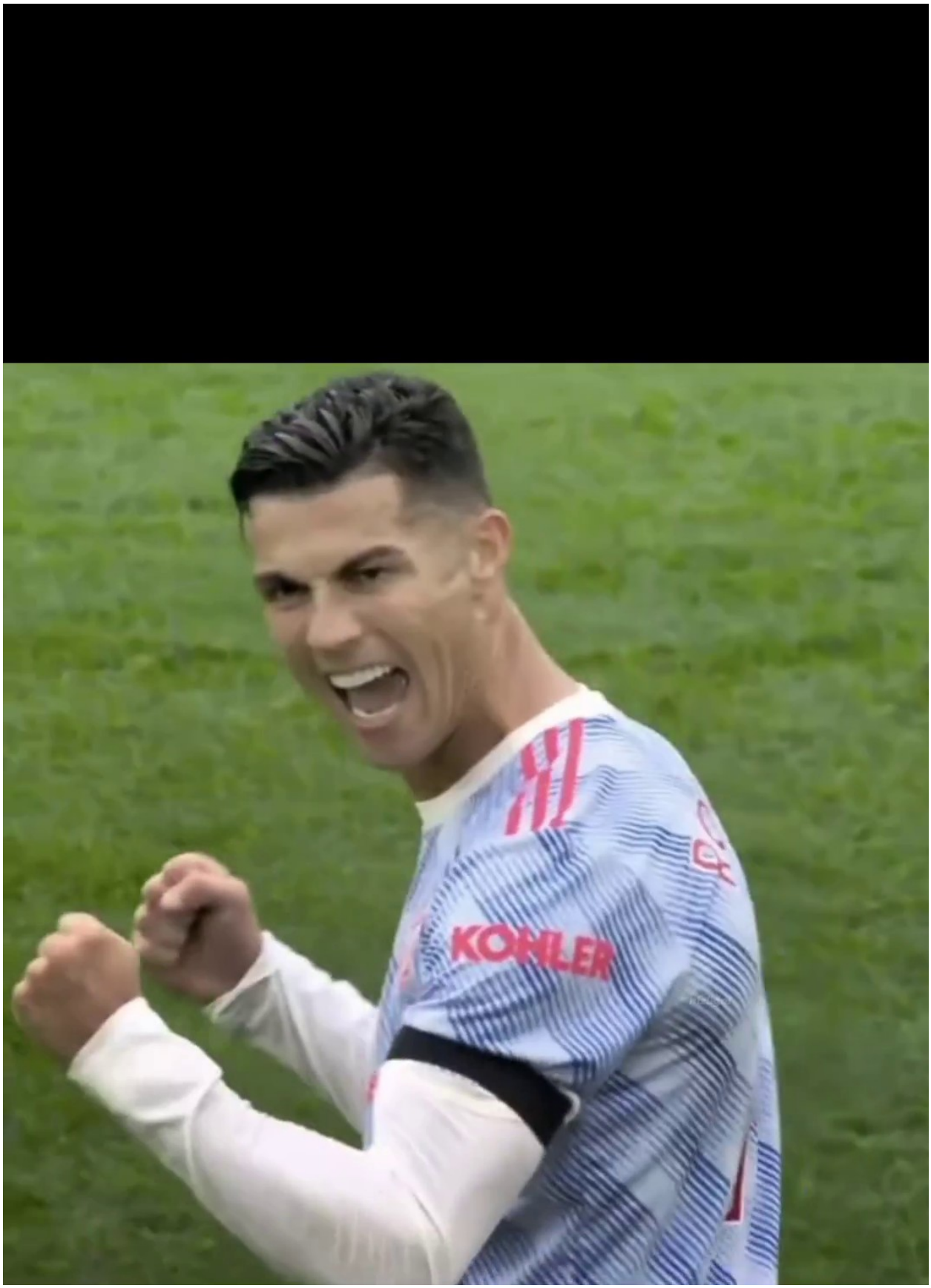
```
0.15366512345678984
Change detected
```


KOHLER

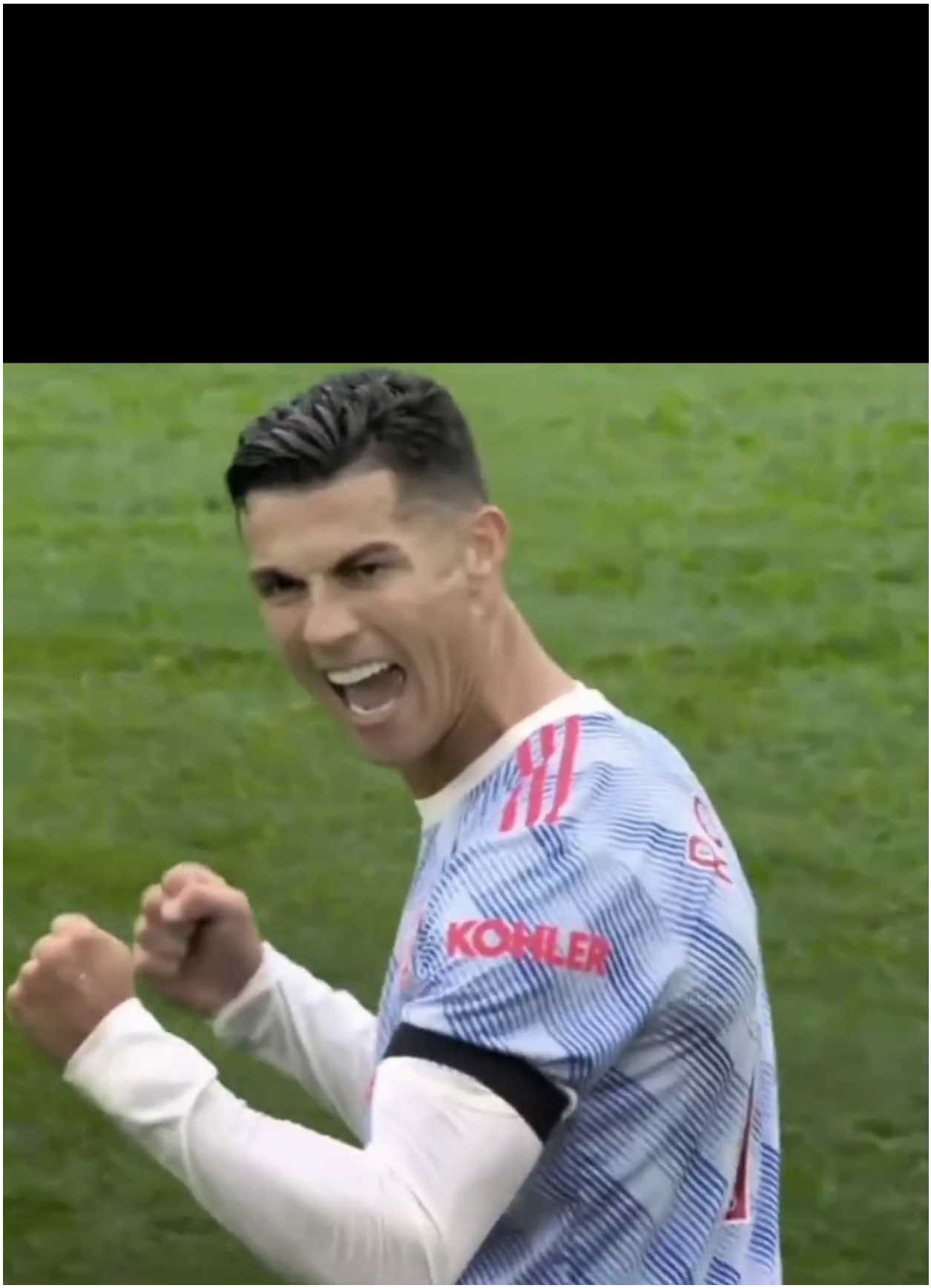
```
0.08837191358024654
Change detected
```


KOHLER

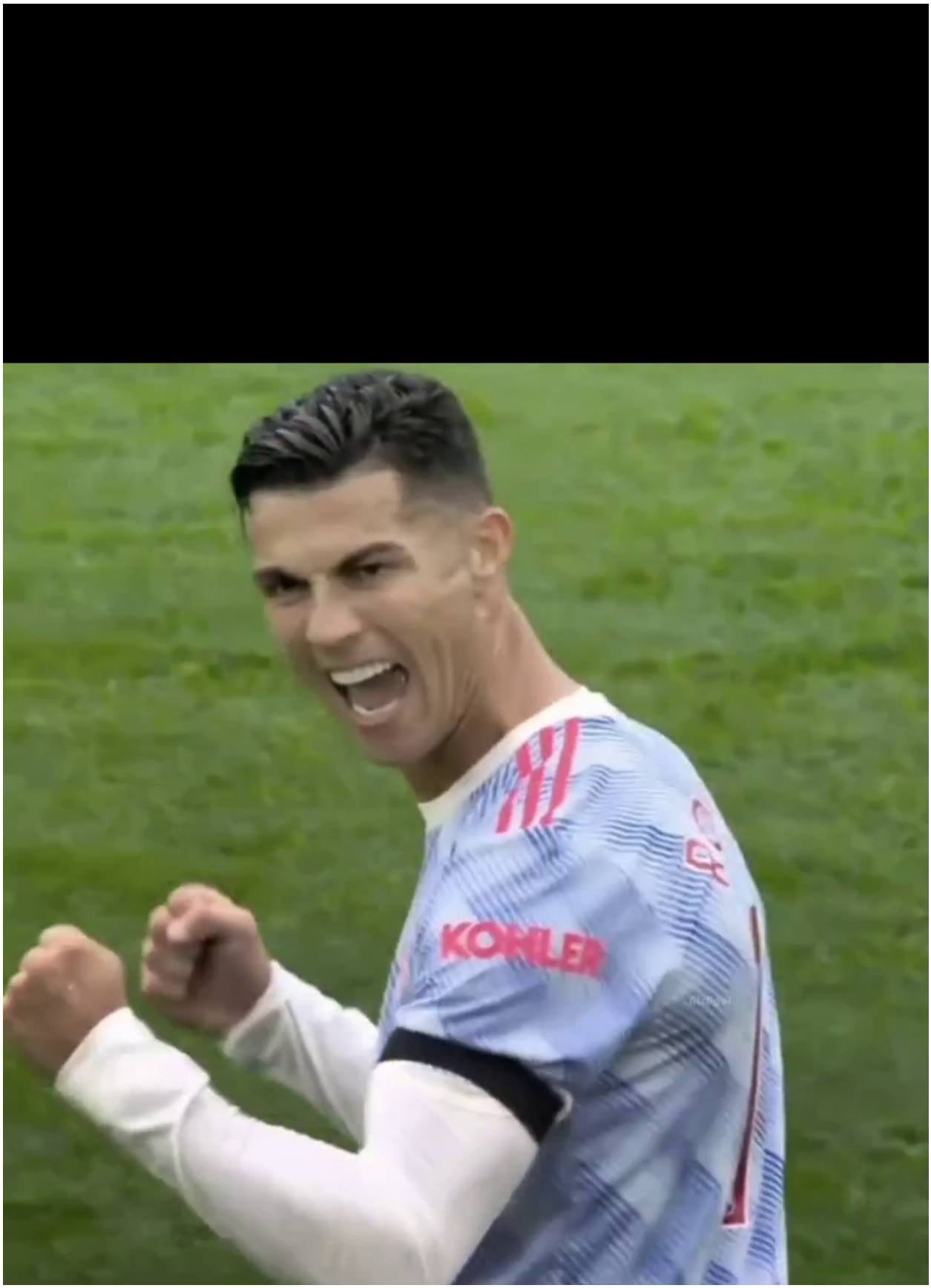
```
0.006894290123454994
Change detected
```


KOHLER

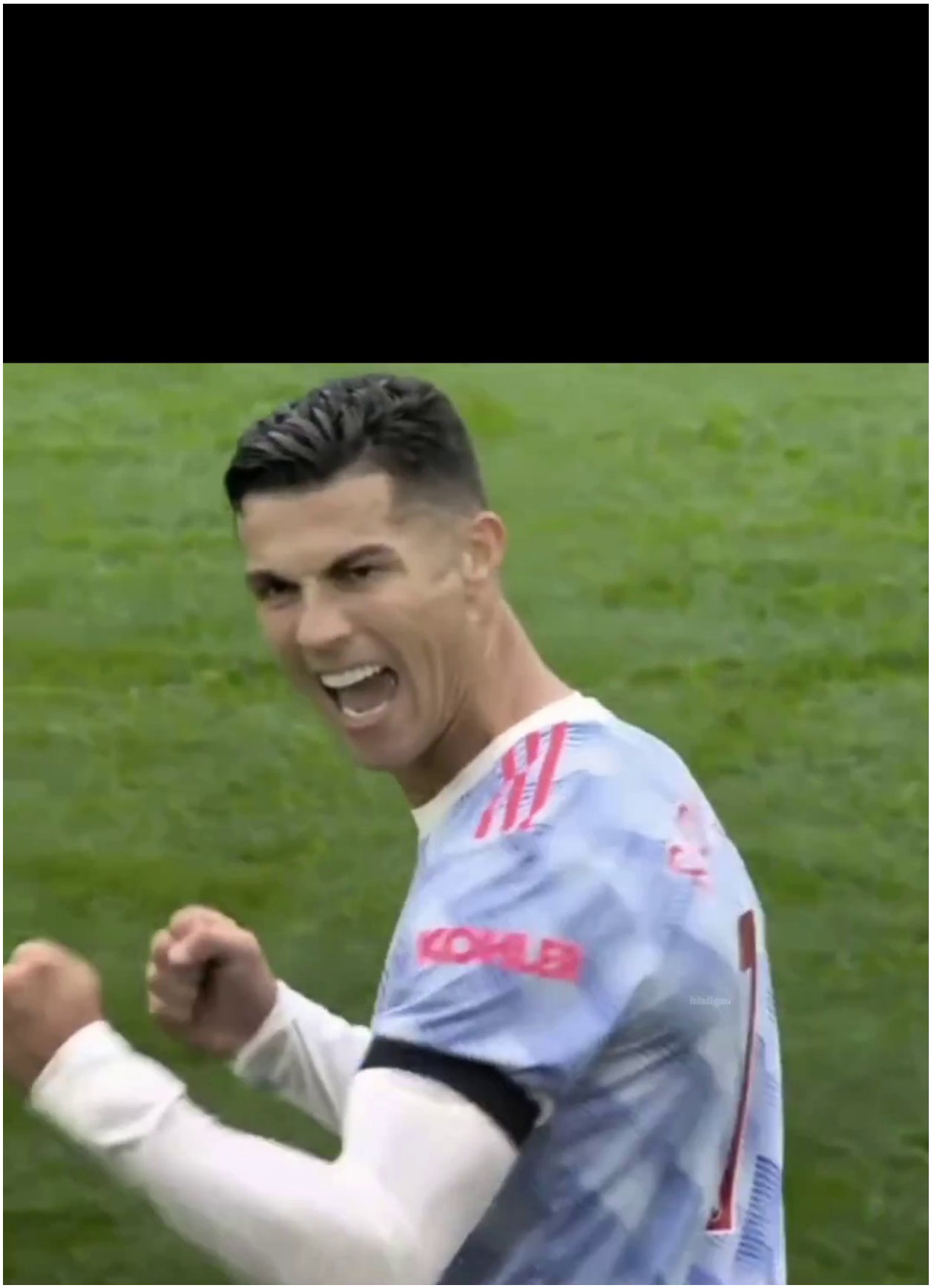
```
0.03742717978394694
Change detected
```


KOHLER

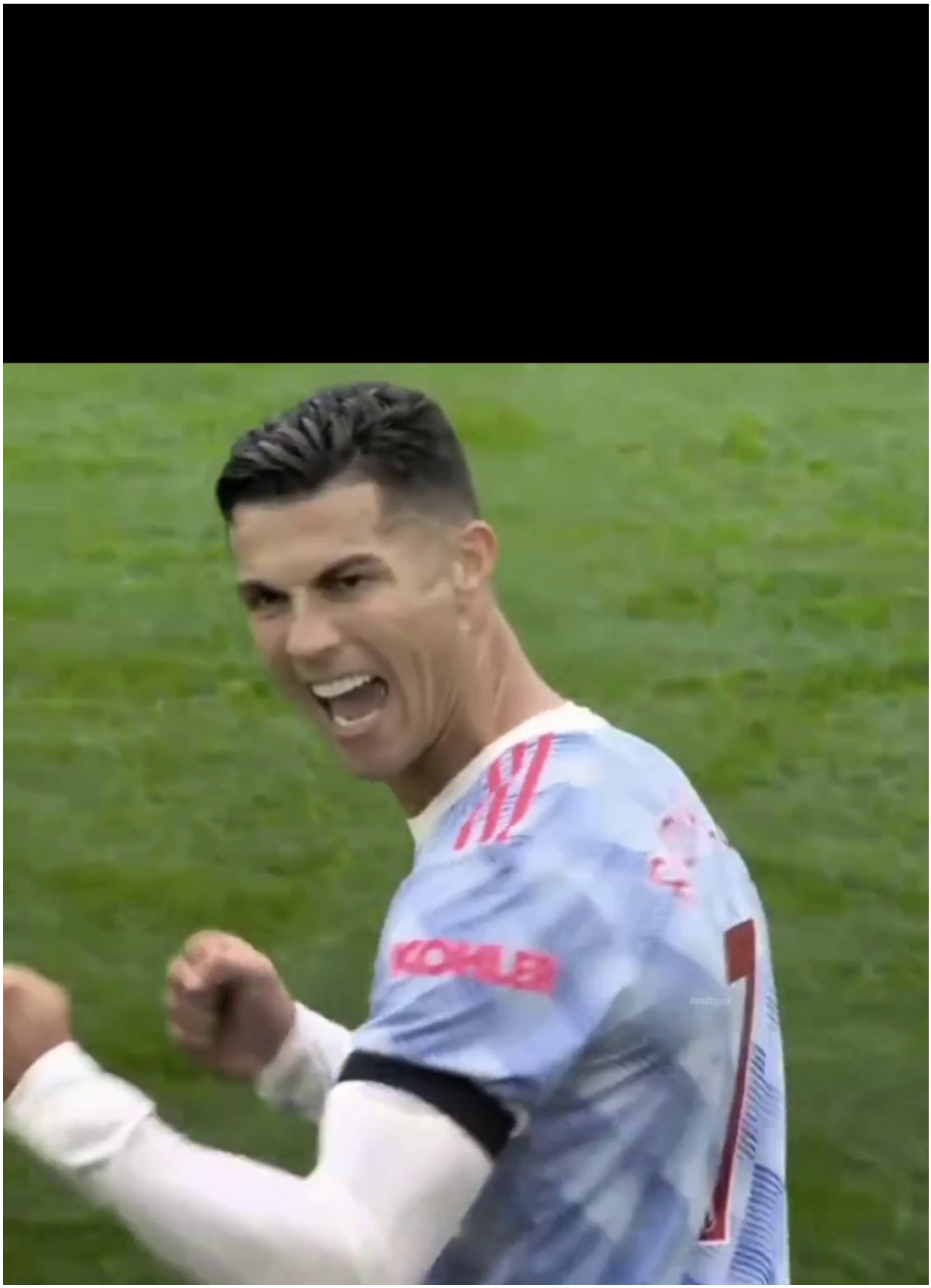
```
0.10041087962963502
Change detected
```


KOHLER

```
0.22392216435184764
Change detected
```



```
0.37755642361112507
Change detected
```



```
0.10220920138890222
Change detected
```


RONALDO
7
hlzligol

```
19.574018614969127
Change detected
```


RONALDO
7
hlzligol

```
0.24344232253086773
Change detected
```


RONALDO
7
hlzligol

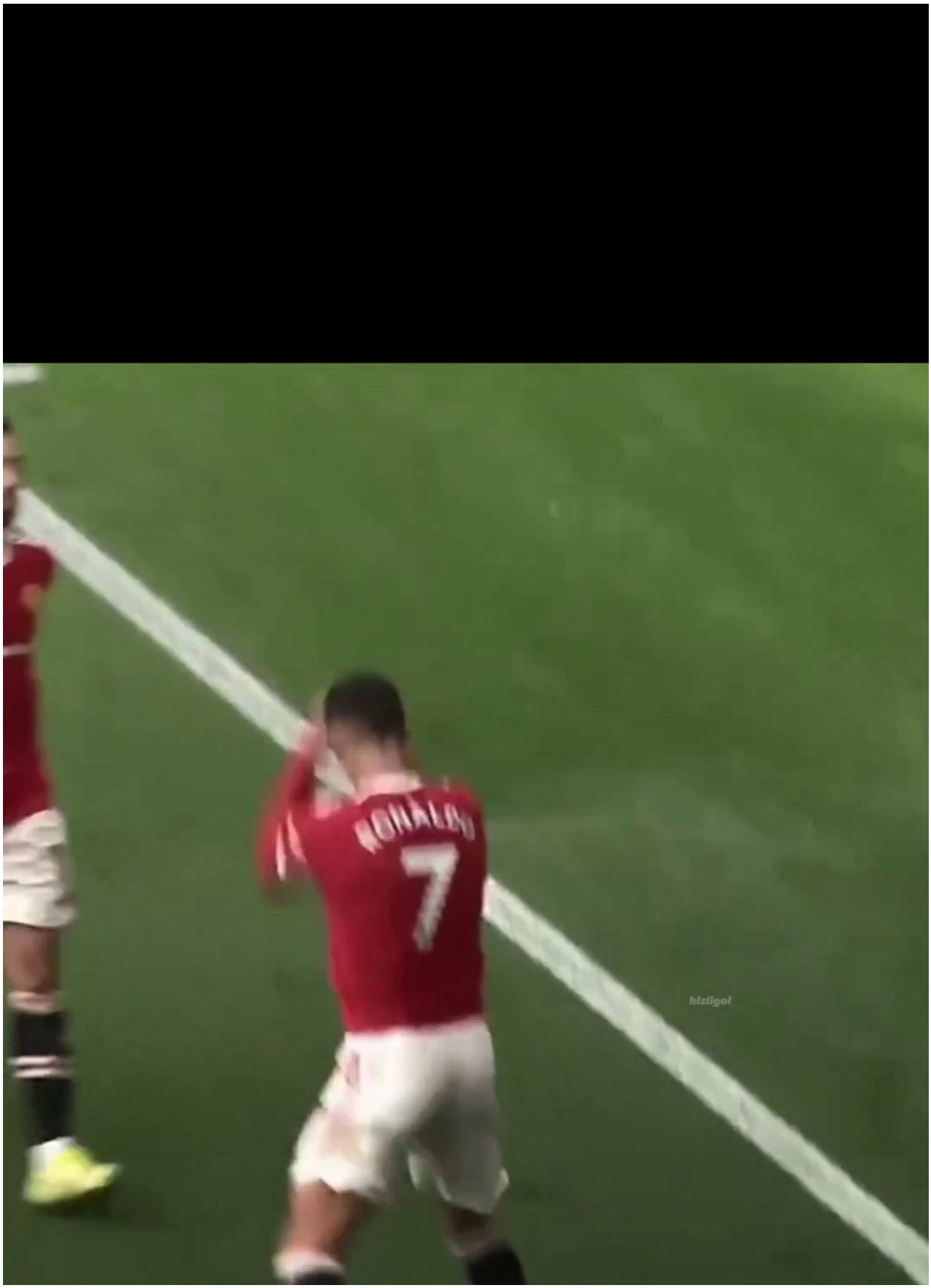
```
0.022759934413578264
Change detected
```


RONALDO
7
hlzligol

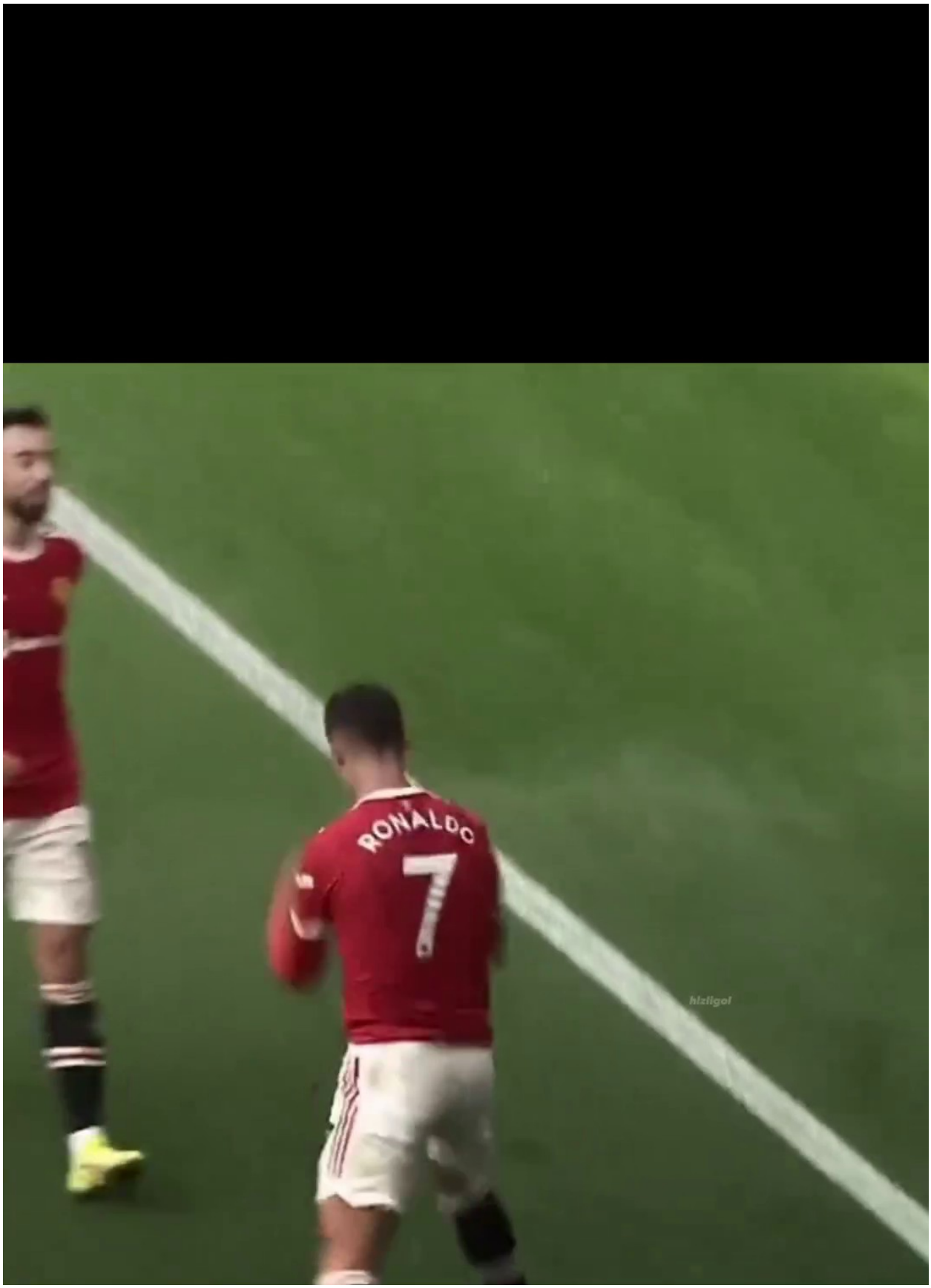
```
0.013610146604939644
Change detected
```


RONALDO
7
hizligol

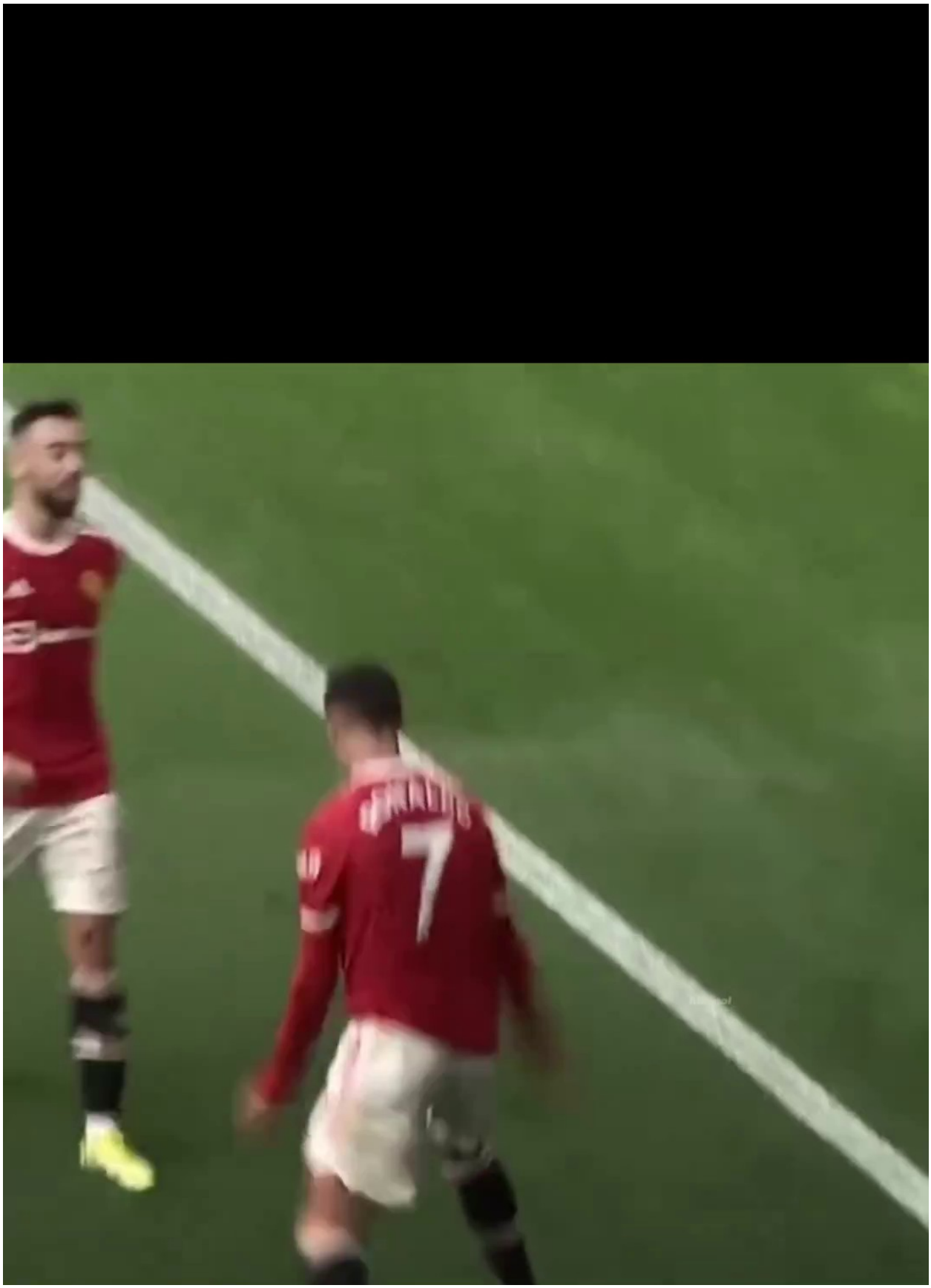
```
0.5110966435185134
Change detected
```


RONALDO
7
hizligol

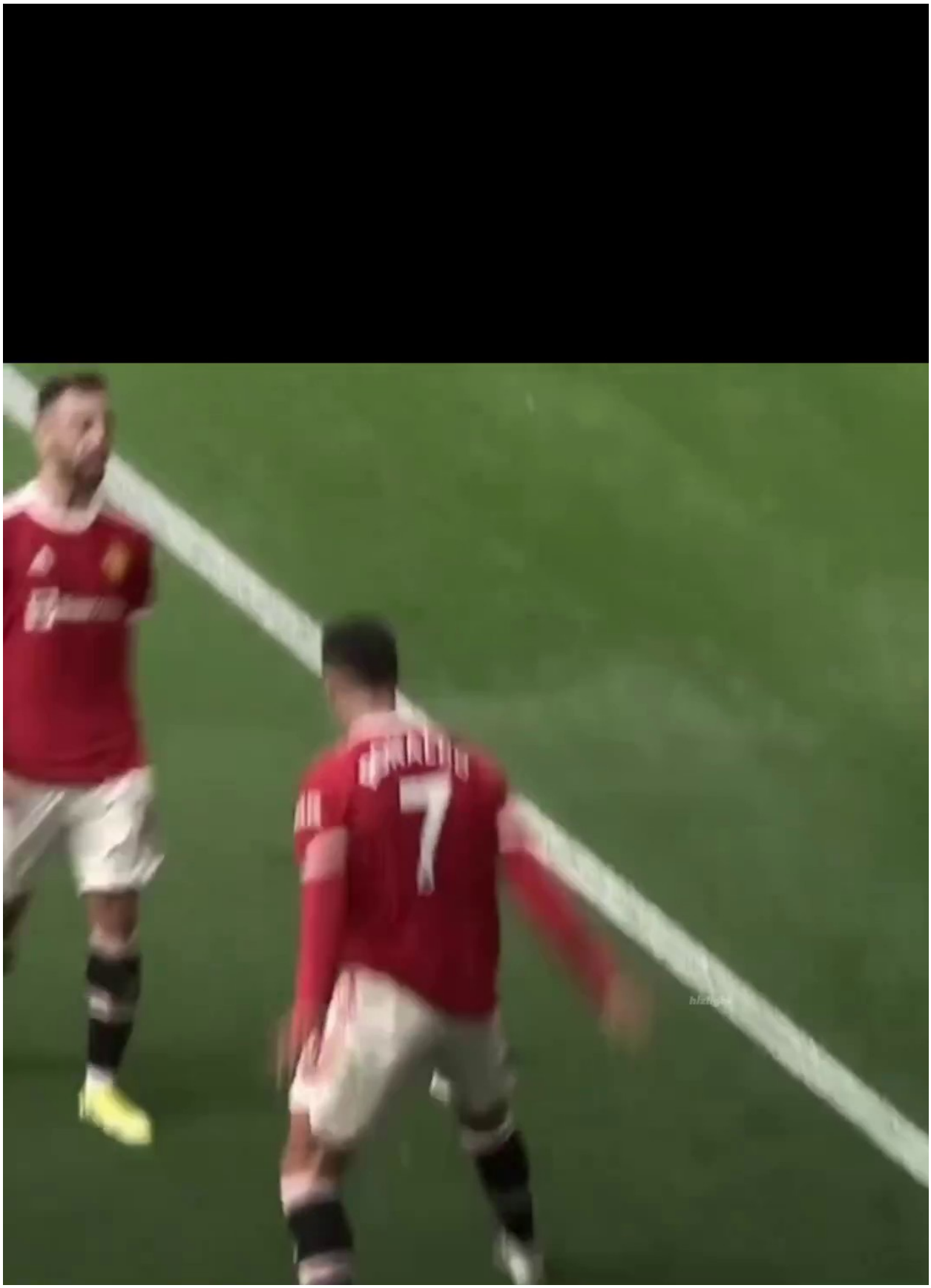
```
0.15378327546296333
Change detected
```


RONALDO
7
hizligol

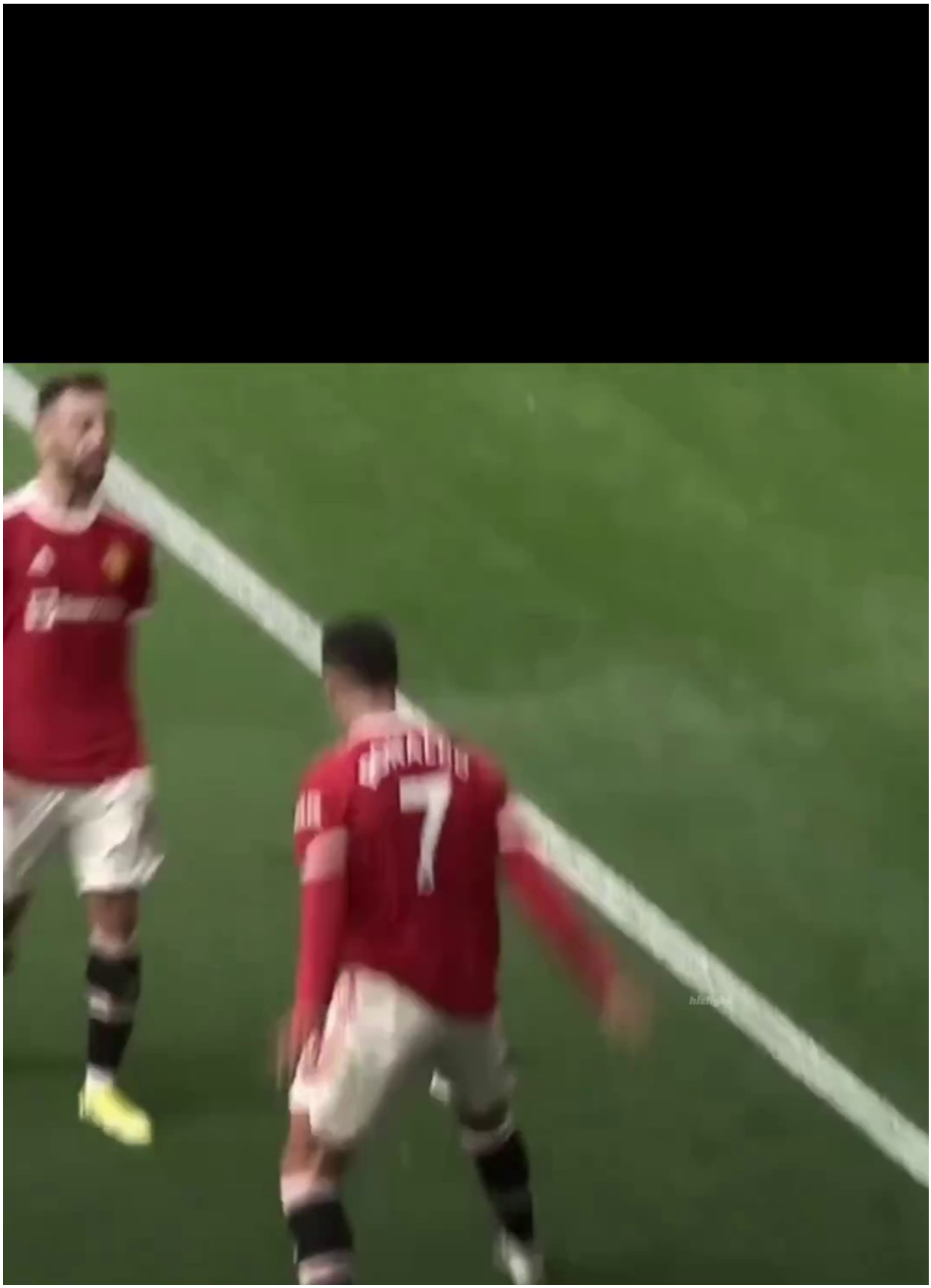
```
0.8563551311728332
Change detected
```



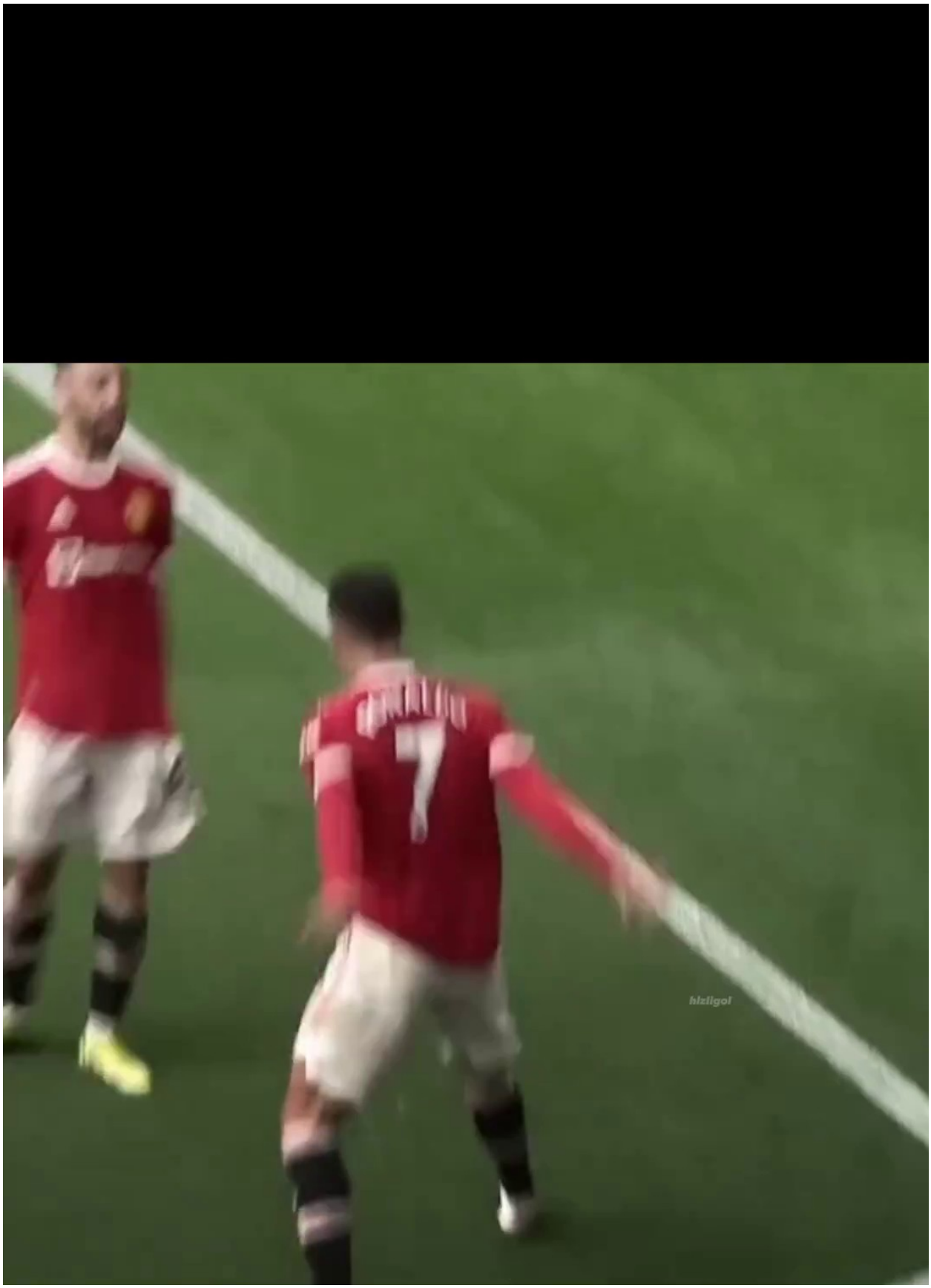
```
0.14303915895062147
Change detected
```


7

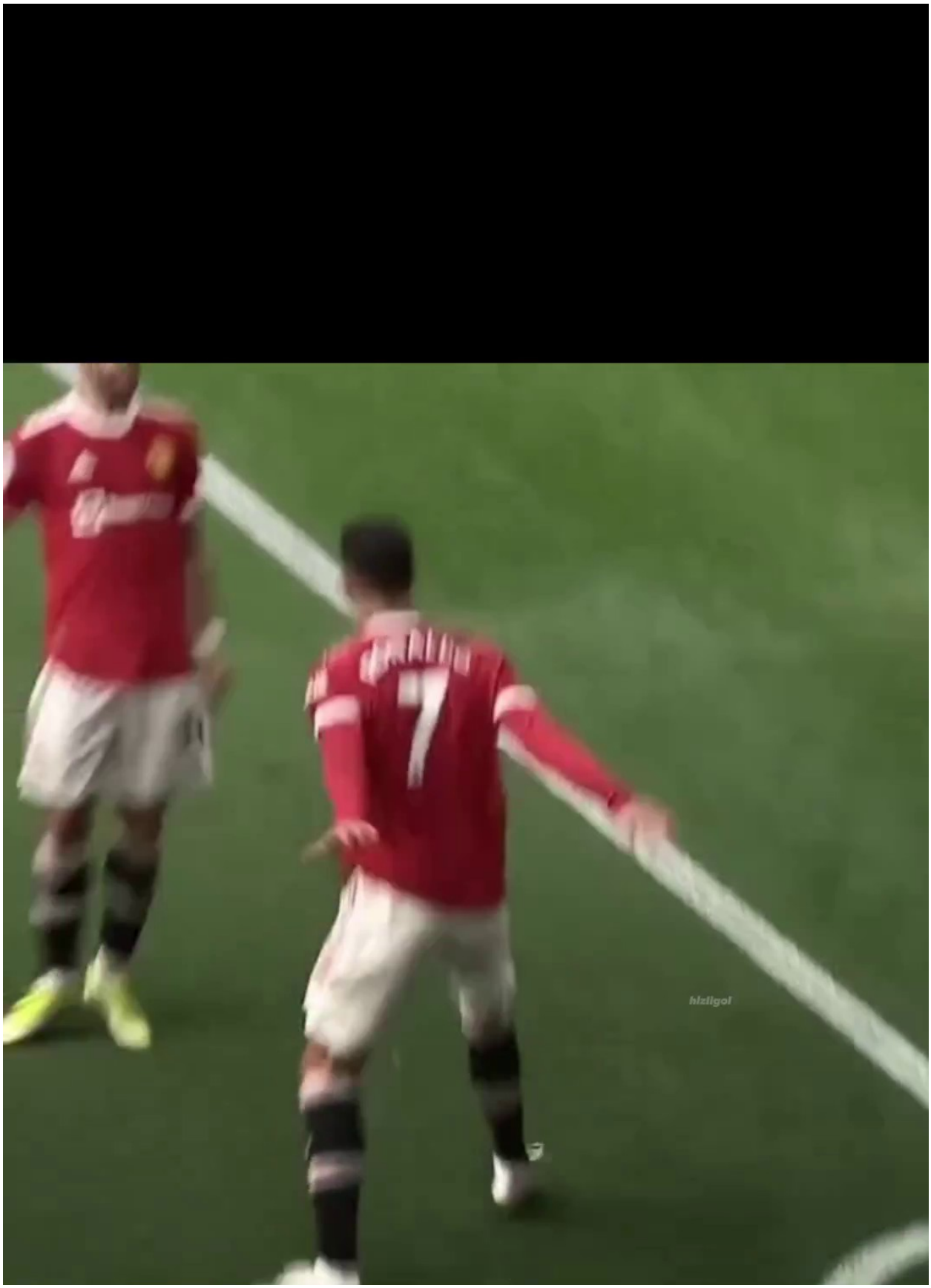
```
0.2883159722222217
Change detected
```



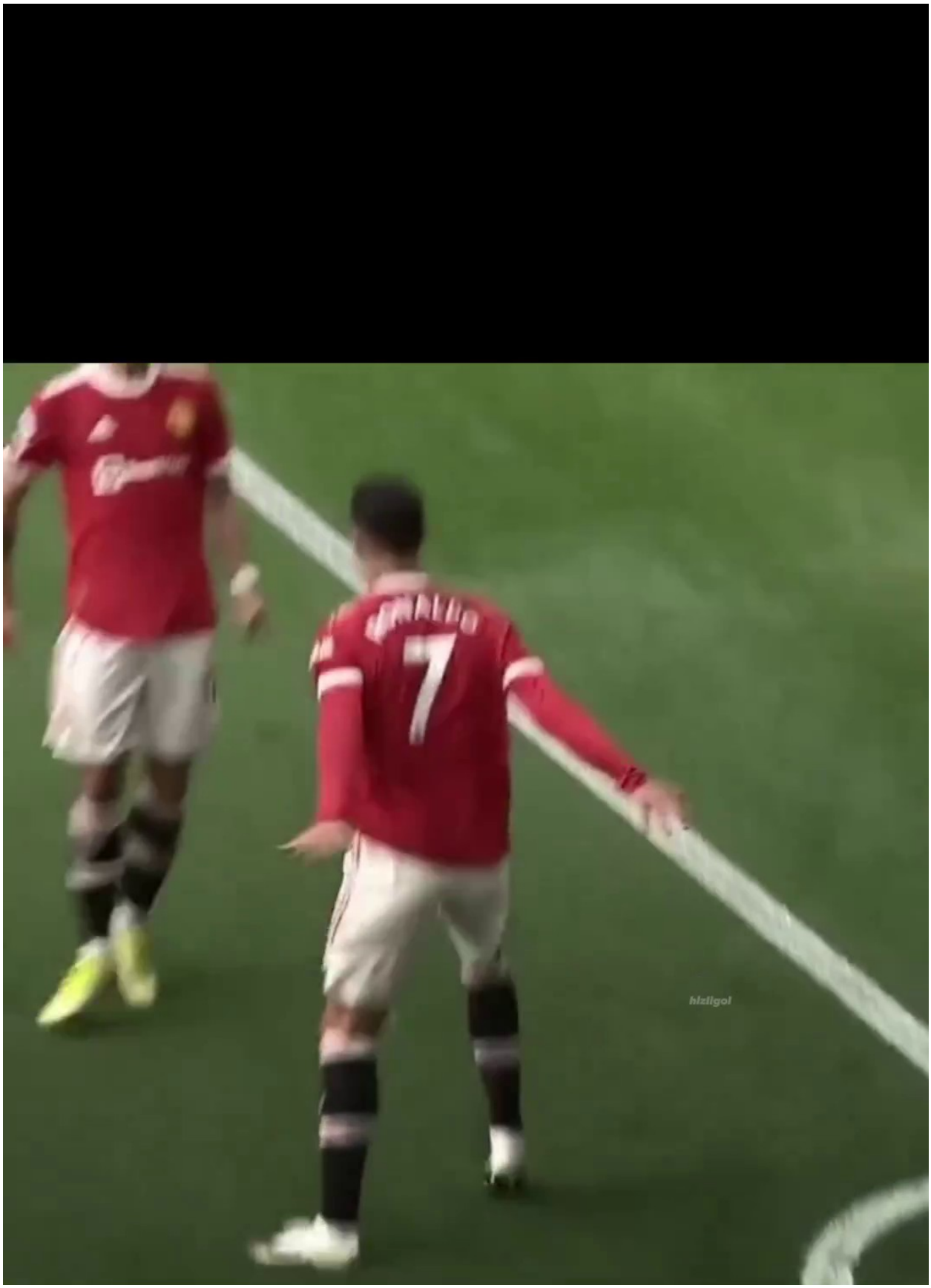
```
0.02205825617284063
Change detected
```


hizligol

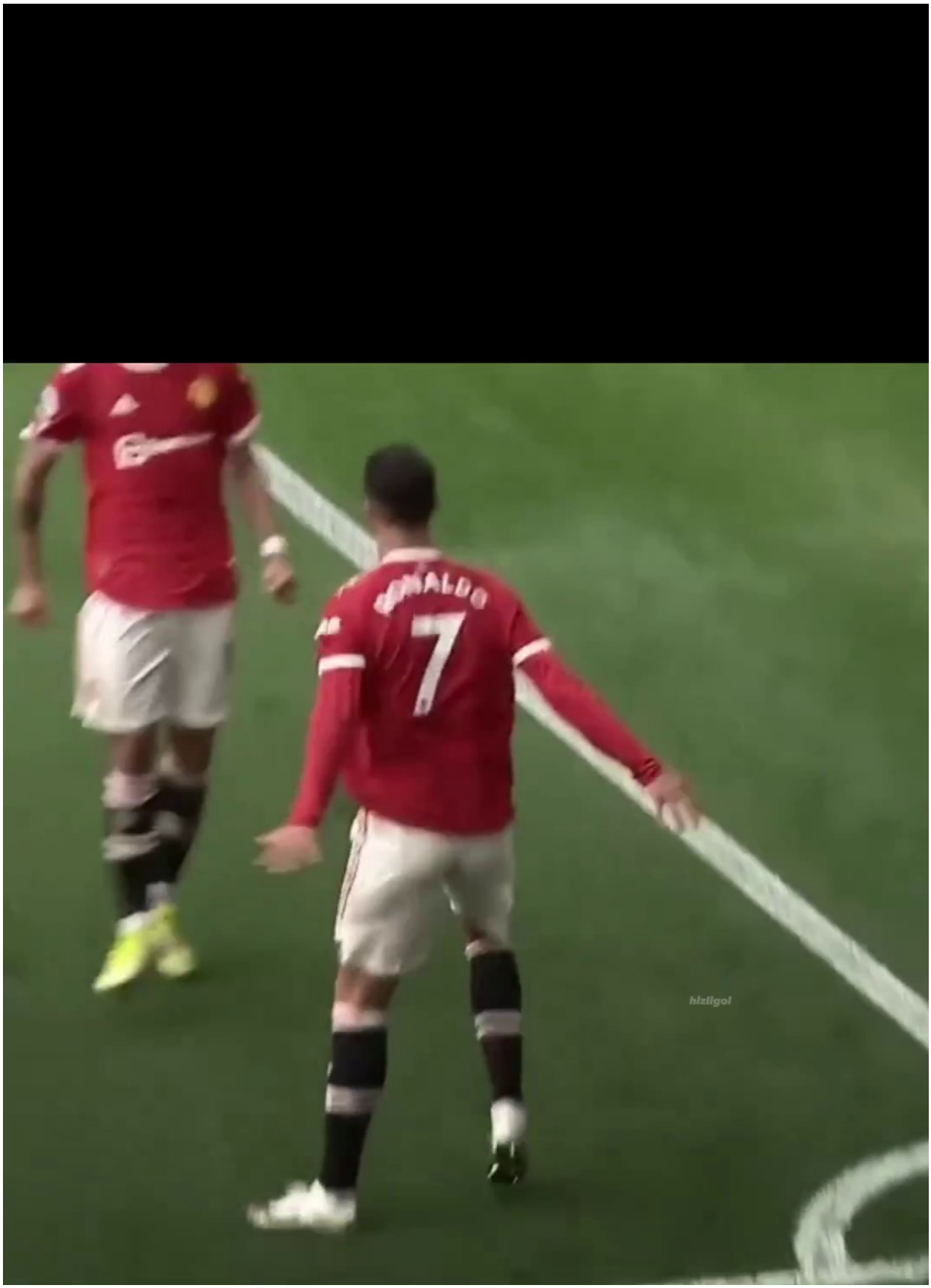
```
0.12366560570987417
Change detected
```


hizligol

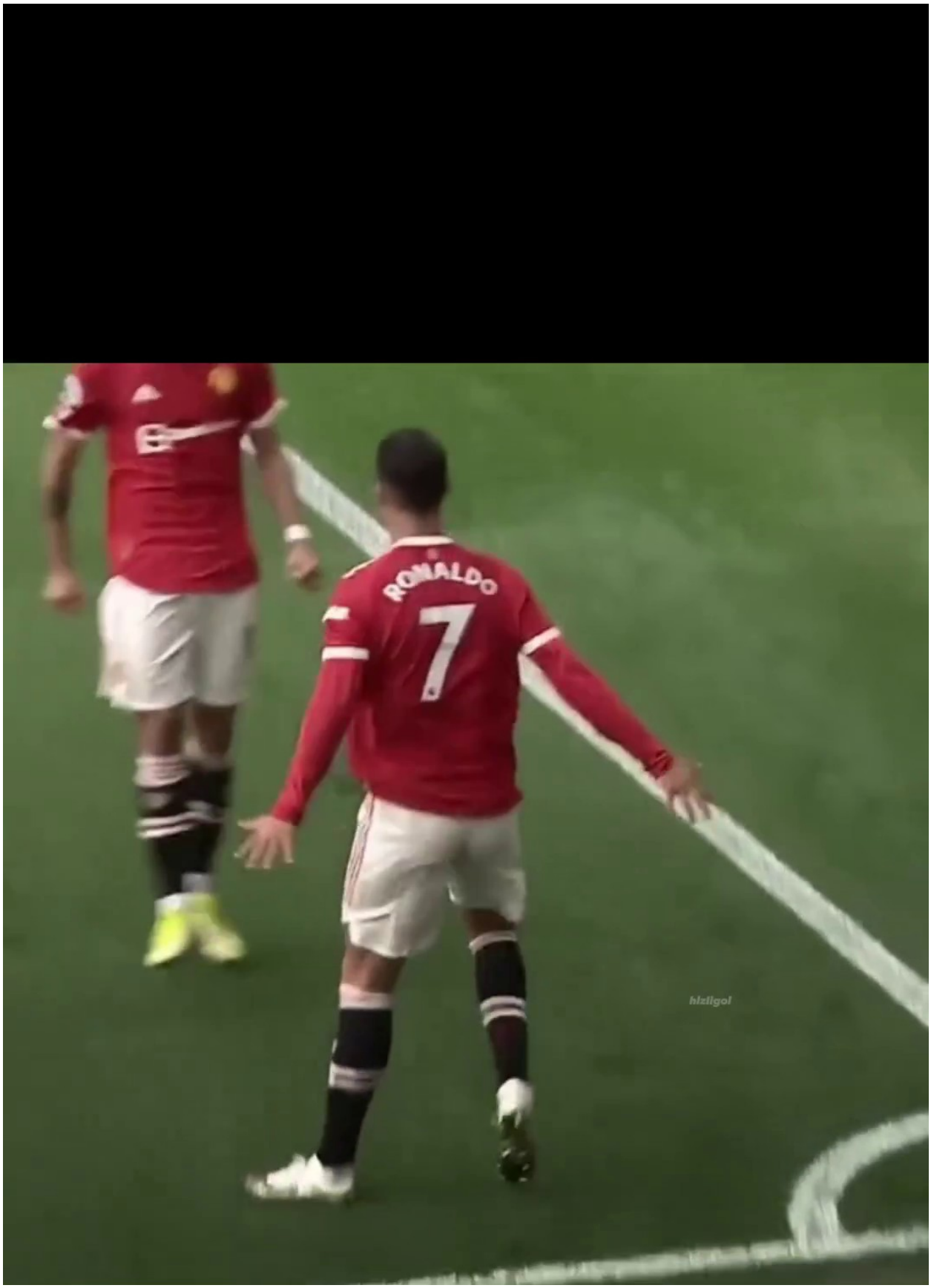
```
0.16315923996913995
Change detected
```


7
hizligol

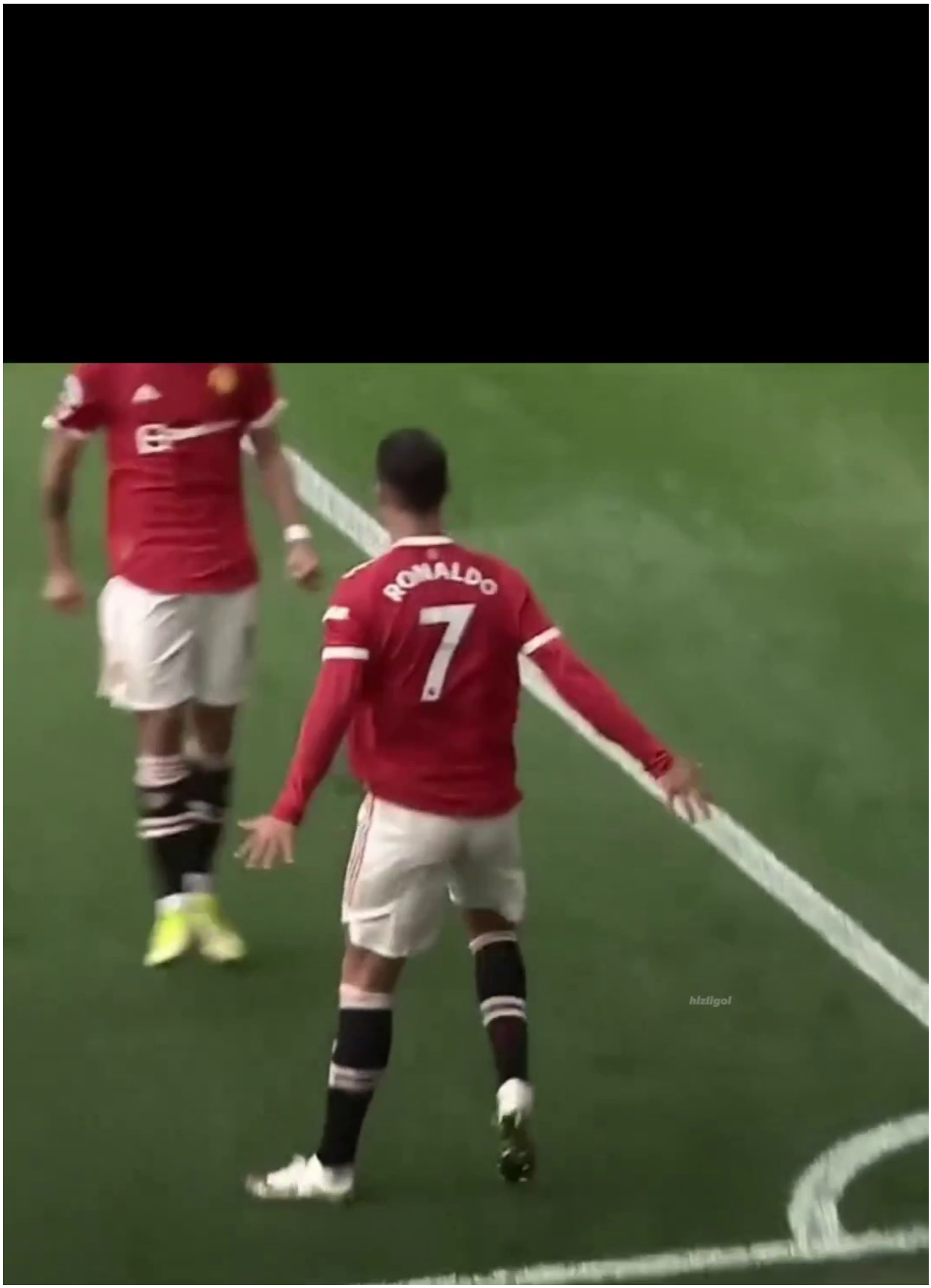
```
0.2567496141975312
Change detected
```


7
hizligol

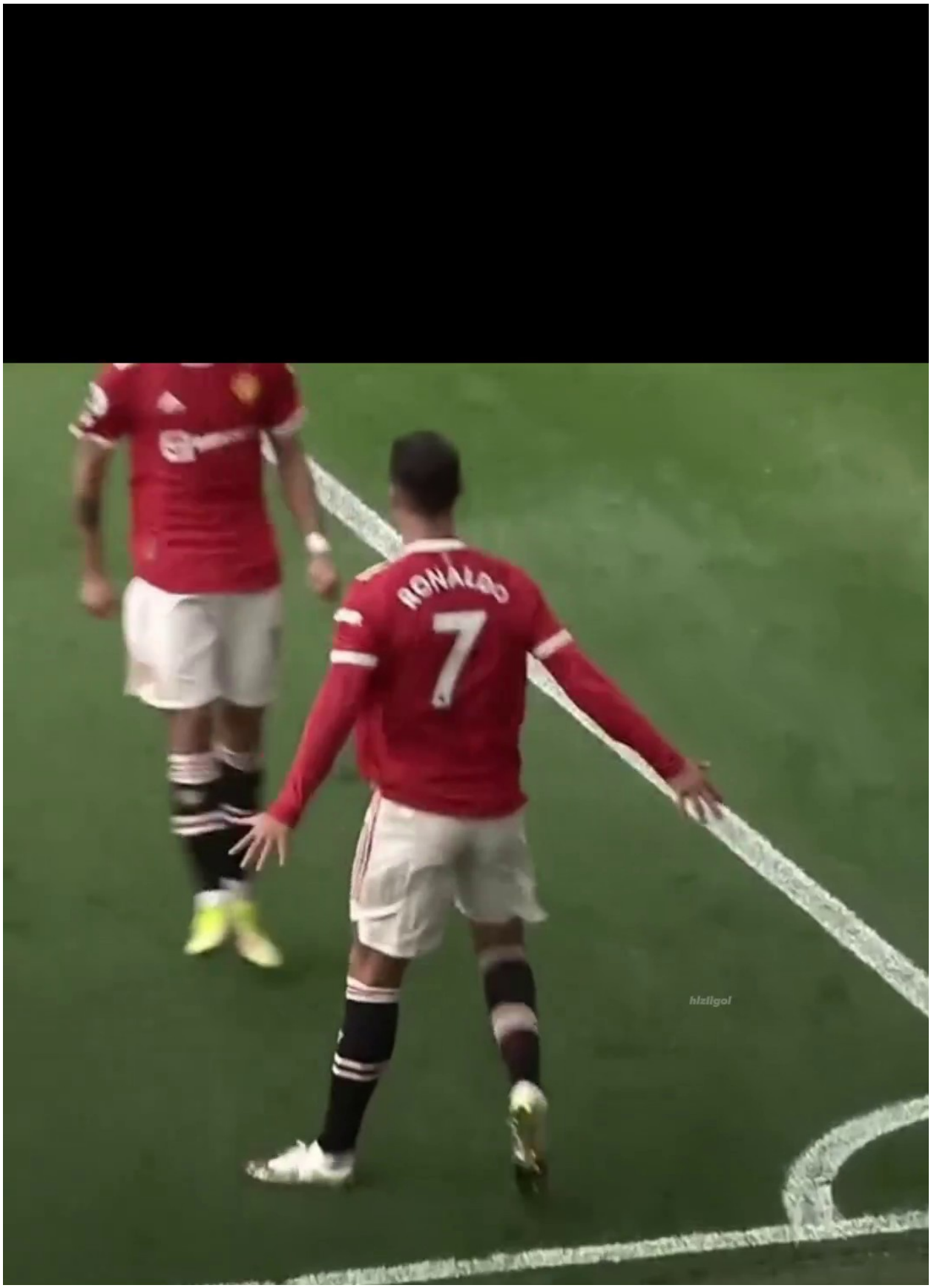
```
0.21685619212962592
Change detected
```


RONALDO
7
hizligol

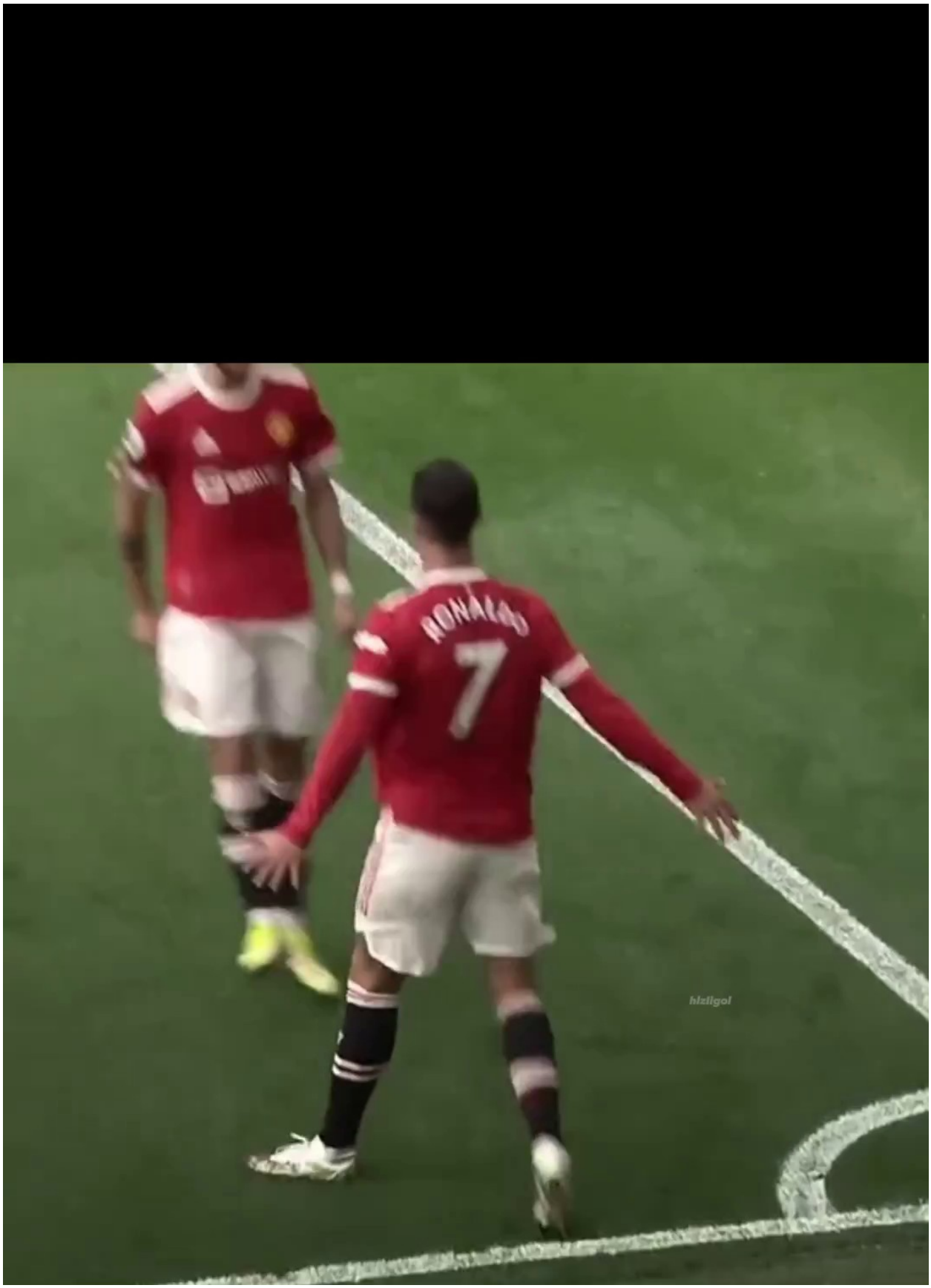
```
0.10874710648148067
Change detected
```


RONALDO
7
hizligol

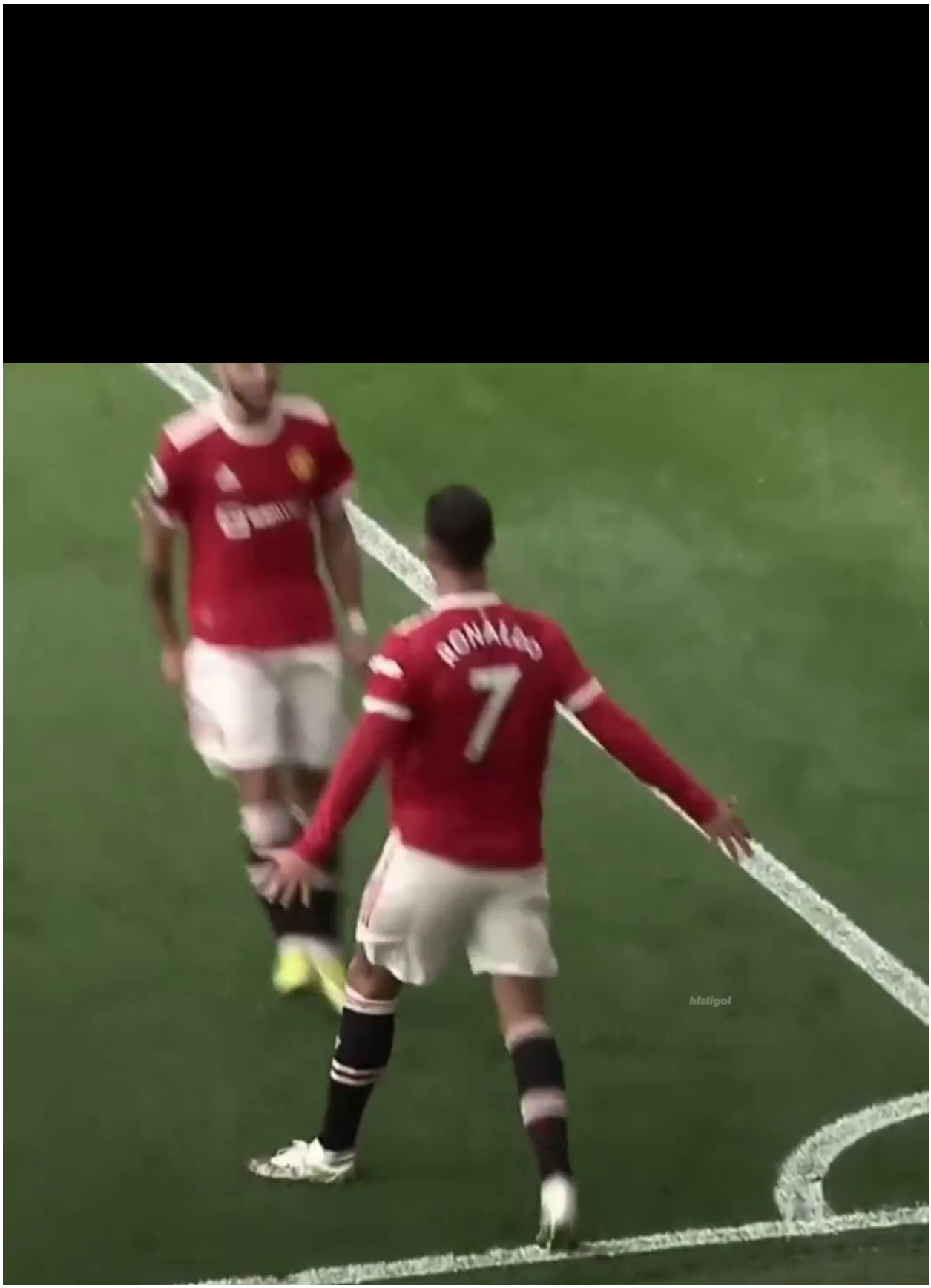
```
0.0020351080246925335
Change detected
```


RONALDO
7
hizligol

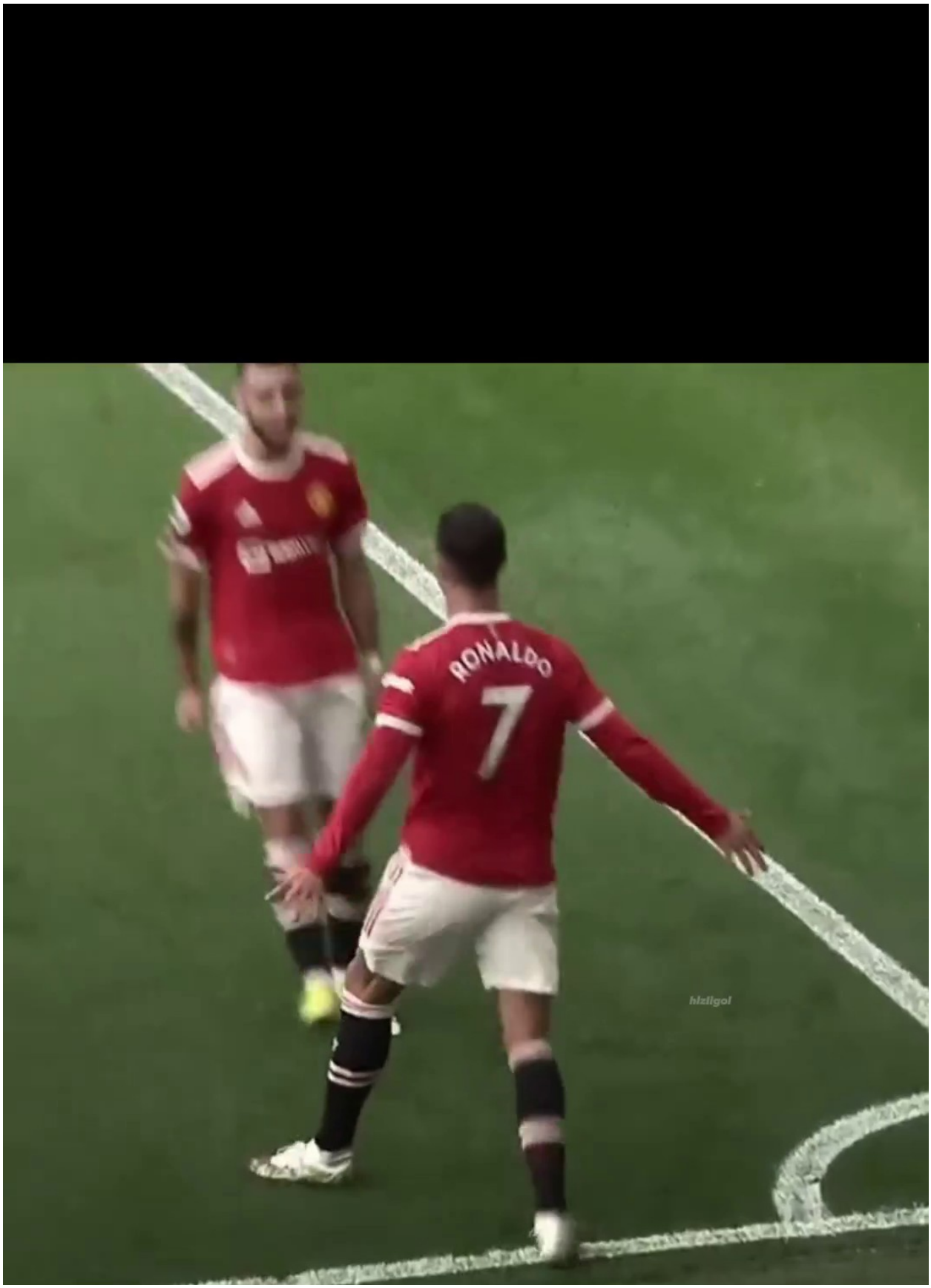
```
0.07265383873456699
Change detected
```


RONALDO
7
hizligol

```
0.5752131558641977
Change detected
```


RONALDO
7
hizligol

```
0.08875385802468827
Change detected
```


RONALDO
7
hizligol

```
#Practical-10-Programs for Face Detection

pip install opencv-python-headless matplotlib
```

```
Requirement already satisfied: opencv-python-headless in
/usr/local/lib/python3.10/dist-packages (4.10.0.82)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: numpy>=1.21.2 in
/usr/local/lib/python3.10/dist-packages (from opencv-python-headless)
(1.25.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib) (1.16.0)
```

```
import urllib.request

# URL of the Haar Cascade file
url =
'https://github.com/opencv/opencv/raw/master/data/haarcascades/haarcas
cade_frontalface_default.xml'

# Path where the file will be saved
haar_cascade_path = 'haarcascade_frontalface_default.xml'

# Download the file
urllib.request.urlretrieve(url, haar_cascade_path)
```

```
('haarcascade_frontalface_default.xml',
 <http.client.HTTPMessage at 0x7e1dbd13a5c0>)
```

```
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

```
import urllib.request

# Download Haar Cascade if not already present
haar_cascade_path = 'haarcascade_frontalface_default.xml'
if not os.path.exists(haar_cascade_path):
    url =
'https://github.com/opencv/opencv/raw/master/data/haarcascades/haarcas
cade_frontalface_default.xml'
    urllib.request.urlretrieve(url, haar_cascade_path)

# Define the image path
img_path = 'Galaxy11.jpg'

# Load the image
img = cv2.imread(img_path)

# Check if the image was loaded properly
if img is None:
    print(f"Error: Could not load image from {img_path}")
else:
    # Convert the image from BGR to RGB
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # Display the original image
    plt.figure(figsize=(8, 6))
    plt.imshow(img_rgb)
    plt.axis('off')
    plt.title('Original Image')
    plt.show()

    # Convert the image to grayscale
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Load the Haar Cascade
    haar_cascade = cv2.CascadeClassifier(haar_cascade_path)

    # Check if the Haar Cascade was loaded properly
    if haar_cascade.empty():
        print(f"Error: Could not load Haar Cascade from
{haar_cascade_path}")
    else:
        # Detect faces
        faces_rect = haar_cascade.detectMultiScale(gray_img,
scaleFactor=1.1, minNeighbors=9)

        # Draw rectangles around detected faces
        for (x, y, w, h) in faces_rect:
            cv2.rectangle(img_rgb, (x, y), (x + w, y + h), (0, 255,
0), 2)
```

```
        # Display the image with detected faces
        plt.figure(figsize=(8, 6))
        plt.imshow(img_rgb)
        plt.axis('off')
        plt.title('Image with Detected Faces')
        plt.show()
```

Image with Detected Faces

```
#Pracrical-11_Object_Detection
#Note-Perform it in Google Collab because the belloe libraries has no
support in Jupyter Notebook

# Step 1: Install Required Libraries
!pip install ultralytics
!pip install matplotlib
!pip install opencv-python-headless

# Step 2: Import Libraries and Load the Model
import matplotlib.pyplot as plt
from ultralytics import YOLO
import cv2

# Load the YOLOv8 model
model = YOLO('yolov8n.pt')

# Step 3: Perform Object Detection and Save Results
# Perform object detection on the input image
image_path = '/content/Emue.jpg'
results = model(image_path)

# Iterate over the results and extract information
for i, result in enumerate(results):
    # Display and save each result
    result.show()
    result_image_path = f'result_{i}.jpg'
    result.save(result_image_path)

# Step 4: Display Original and Result Image Side by Side
# Load the original image
original_image = cv2.imread(image_path)
original_image_rgb = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)

# Load and display all result images
plt.figure(figsize=(20, 10))

plt.subplot(1, 2, 1)
plt.imshow(original_image_rgb)
plt.axis('on')
plt.title('Original Image')

for i in range(len(results)):
    result_image = cv2.imread(f'result_{i}.jpg')
    result_image_rgb = cv2.cvtColor(result_image, cv2.COLOR_BGR2RGB)
    plt.subplot(1, 2, 2)
    plt.imshow(result_image_rgb)
    plt.axis('on')
    plt.title('YOLOv8 Object Detection Result')
```

```
plt.show()
```

```
Requirement already satisfied: ultralytics in
/usr/local/lib/python3.10/dist-packages (8.2.35)
Requirement already satisfied: numpy<2.0.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (1.25.2)
Requirement already satisfied: matplotlib>=3.3.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (3.7.1)
Requirement already satisfied: opencv-python>=4.6.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (4.8.0.76)
Requirement already satisfied: pillow>=7.1.2 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (9.4.0)
Requirement already satisfied: pyyaml>=5.3.1 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (6.0.1)
Requirement already satisfied: requests>=2.23.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (2.31.0)
Requirement already satisfied: scipy>=1.4.1 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (1.11.4)
Requirement already satisfied: torch>=1.8.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics)
(2.3.0+cu121)
Requirement already satisfied: torchvision>=0.9.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics)
(0.18.0+cu121)
Requirement already satisfied: tqdm>=4.64.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (4.66.4)
Requirement already satisfied: psutil in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: py-cpuinfo in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: pandas>=1.1.4 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (2.0.3)
Requirement already satisfied: seaborn>=0.11.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (0.13.1)
Requirement already satisfied: ultralytics-thop>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (2.0.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (1.2.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (24.1)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4-
>ultralytics) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4-
>ultralytics) (2024.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.23.0-
>ultralytics) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.23.0-
>ultralytics) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.23.0-
>ultralytics) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.23.0-
>ultralytics) (2024.6.2)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (3.14.0)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (4.12.2)
Requirement already satisfied: sympy in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (1.12.1)
Requirement already satisfied: networkx in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (3.3)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (3.1.4)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (2023.6.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (12.1.105)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105
in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (12.1.105)
```

```
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (12.1.105)
Requirement already satisfied: nvidia-cudnn-cu12==8.9.2.26 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (8.9.2.26)
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (12.1.3.1)
Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (11.0.2.54)
Requirement already satisfied: nvidia-curand-cu12==10.3.2.106 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (10.3.2.106)
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (11.4.5.107)
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (12.1.0.106)
Requirement already satisfied: nvidia-nccl-cu12==2.20.5 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (2.20.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (12.1.105)
Requirement already satisfied: triton==2.3.0 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (2.3.0)
Requirement already satisfied: nvidia-nvjitlink-cu12 in
/usr/local/lib/python3.10/dist-packages (from nvidia-cusolver-
cu12==11.4.5.107->torch>=1.8.0->ultralytics) (12.5.40)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib>=3.3.0->ultralytics) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.8.0-
>ultralytics) (2.1.5)
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.8.0-
>ultralytics) (1.3.0)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.53.0)
```

```
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.25.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib) (1.16.0)
Requirement already satisfied: opencv-python-headless in
/usr/local/lib/python3.10/dist-packages (4.10.0.82)
Requirement already satisfied: numpy>=1.21.2 in
/usr/local/lib/python3.10/dist-packages (from opencv-python-headless)
(1.25.2)

image 1/1 /content/Emue.jpg: 480x640 1 bicycle, 1 car, 1 truck, 1 dog,
204.5ms
Speed: 4.0ms preprocess, 204.5ms inference, 1.2ms postprocess per
image at shape (1, 3, 480, 640)
```