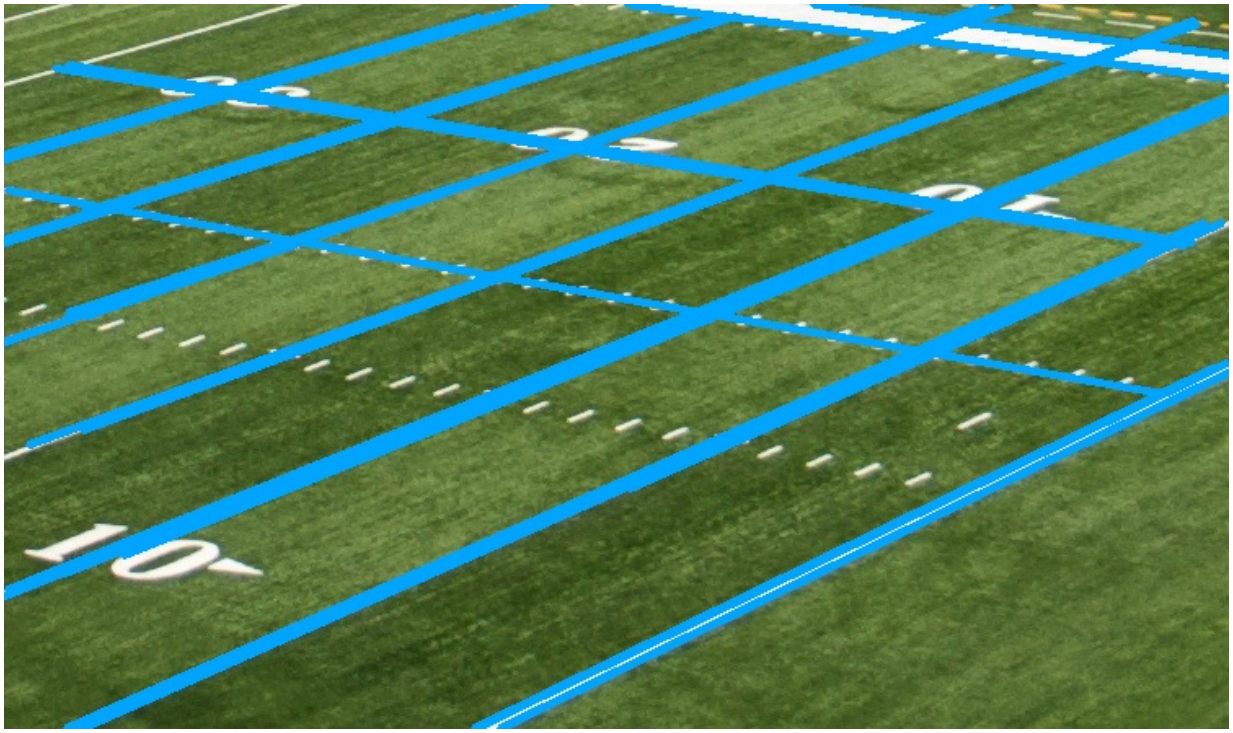
In [2]:
```
import numpy as np
import pandas as pd
import cv2
import math
```

Hough Transformation

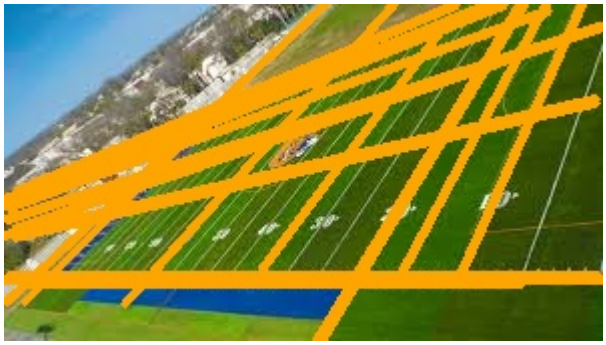
In [13]:
```
# Read image
img = cv2.imread('campo1.jpg', cv2.IMREAD_COLOR) # road.png is the filename
# Convert the image to gray-scale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Find the edges in the image using canny detector
edges = cv2.Canny(gray, 50, 200)
# Detect points that form a line
lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold=120, minLineLength=10, ma
# Draw lines on the image
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(img, (x1, y1), (x2, y2), (255,165,0), 3)
# Show result
cv2.imwrite("h1.jpg", img)
```

Out[13]: True



In [2]:
```
# Read image
img = cv2.imread('campo2.jpg', cv2.IMREAD_COLOR) # road.png is the filename
# Convert the image to gray-scale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Find the edges in the image using canny detector
edges = cv2.Canny(gray, 50, 200)
# Detect points that form a line
lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold=100, minLineLength=10, ma
# Draw lines on the image
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(img, (x1, y1), (x2, y2), (0,165,255), 3)
# Show result
cv2.imwrite("h2.jpg", img)
```

Out[2]: True



In [22]:
```
# Read image
img = cv2.imread('campo3.jpg', cv2.IMREAD_COLOR) # road.png is the filename
# Convert the image to gray-scale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Find the edges in the image using canny detector
edges = cv2.Canny(gray, 50, 200)
# Detect points that form a line
lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold=120, minLineLength=10, ma
# Draw lines on the image
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(img, (x1, y1), (x2, y2), (0,165,255), 2)
# Show result
cv2.imwrite("h3.jpg", img)
```

Out[22]: True

Defining the Hough transformation function

In [24]:
```
def hough_trans(image_in, image_out,thresh=100, color=(0,0,0), line_thickness=3)
    # Read image
    img = cv2.imread(image_in, cv2.IMREAD_COLOR) # road.png is the filename
    # Convert the image to gray-scale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Find the edges in the image using canny detector
    edges = cv2.Canny(gray, 50, 200)
    # Detect points that form a line
    lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold=thresh, minLineLength=
    # Draw lines on the image
    for line in lines:
        x1, y1, x2, y2 = line[0]
        cv2.line(img, (x1, y1), (x2, y2), color, 3)
    # Show result
    cv2.imwrite(image_out, img)
```

In [25]:
```
hough_trans("campo1.jpg","test.jpg")
```

K means for image segmentation

In [3]:
```
img = cv2.imread('frutas.jpg',cv2.IMREAD_COLOR)
Z = img.reshape((-1,3))
Z = np.float32(Z)
# define criteria, number of clusters(K) and apply kmeans()
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
K = 10
ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)

# Now convert back into uint8, and make original image
center = np.uint8(center)
res = center[label.flatten()]
res2 = res.reshape((img.shape))
label = label.reshape((img.shape[0],img.shape[1]))
cv2.imshow('res2',res2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [4]:
```
img.shape
```

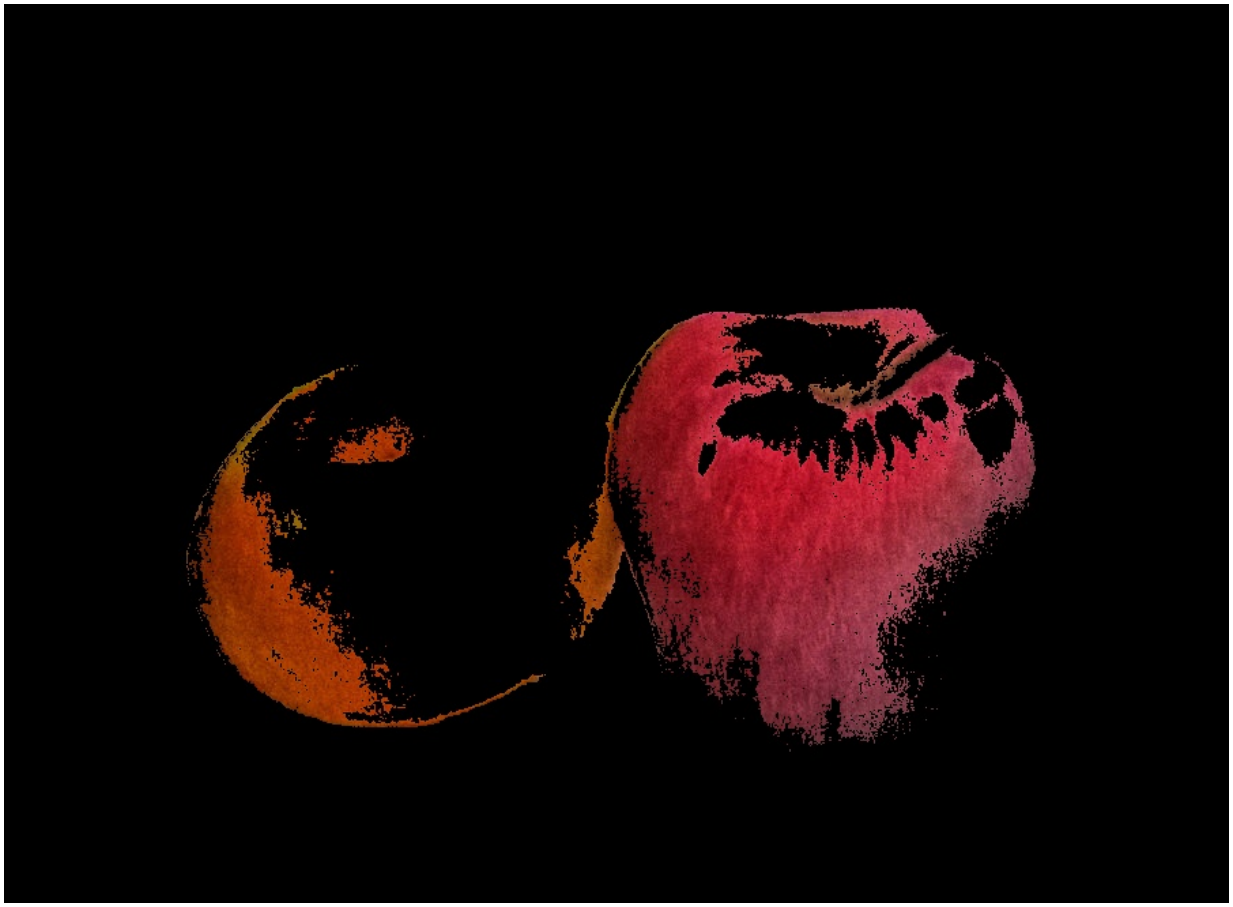
Out[4]:
```
(660, 900, 3)
```

Extracting clusters from segmented image

Apple and orange both share labels from the same cluster

In [5]:
```
def extractCluster(image,label_image,label):
    component=np.zeros(image.shape,np.uint8)
    component[label_image==label]=image[label_image==label]
    return component
```

In [6]:
```
extracted=extractCluster(img,label,0)
extracted.shape
cv2.imwrite('apple.jpg',extracted)
cv2.imshow('res2',extracted)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



In [ ]:
```
```