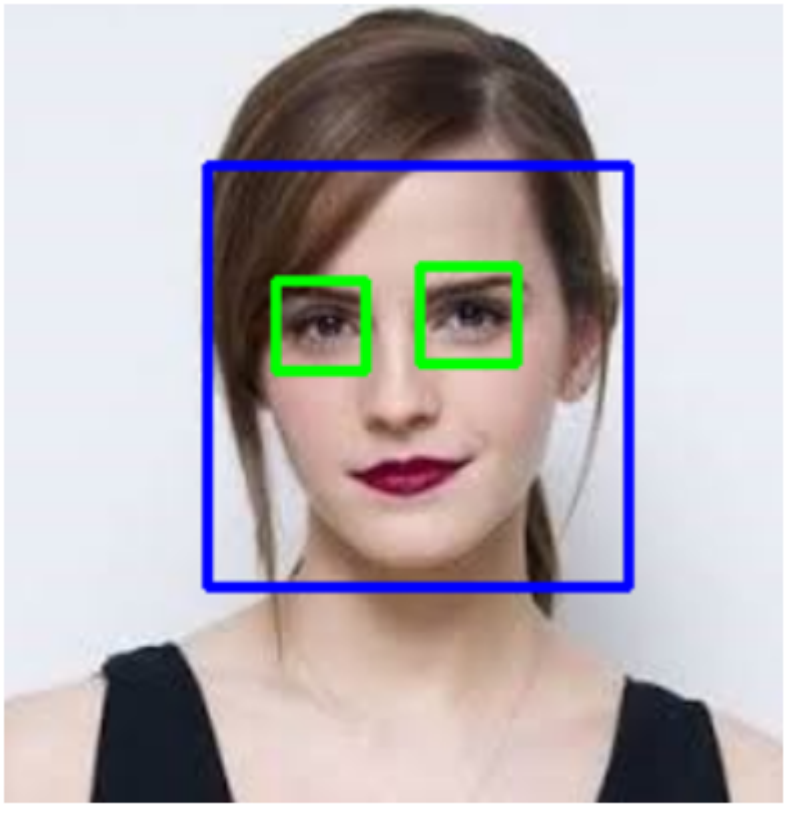


Why Face and Eye Detection?

Face and eye detection are fundamental tasks in computer vision with various practical applications. Here are some reasons why face and eye detection are crucial:

- Biometric Identification/ Accessibility Features/ Facial Recognition Login : Recognizing faces and eyes is essential for security applications such as facial recognition access control systems.
- Photography and Image Editing : Face and eye detection are widely used in photography and image editing software. These technologies help in automatic focusing, red-eye removal, and other enhancements.
- Driver Monitoring Systems : In automotive technology, face and eye detection are employed in driver monitoring systems to ensure that drivers are attentive and alert. This enhances safety on the road.
- Healthcare Applications
- Social Media and Entertainment
- Security and Surveillance
- Emotion Recognition

Importing necessary package

In [1]:
```
import os
import numpy
import pandas as pd
import cv2
import matplotlib.pyplot as plt
```

Haar Cascade classifiers ¶

- What's Haar Cascade?

Haar Cascade classifiers are machine learning object detection methods used to identify objects in images or video. In current project, pre-trained Haar Cascade classifiers has been utilized for detecting faces and eyes .

In [2]:
```
face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_classifier = cv2.CascadeClassifier('haarcascade_eye.xml')
```

Face Detection

- Read the Image: img = cv2.imread('img.jpg')
- Convert Image to Grayscale: cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) Converts the color image (BGR format) to grayscale using the cvtColor function.
- Face Detection: Uses the detectMultiScale method of the face classifier to detect faces in the grayscale image.
- Check if Faces are Found: Checks if any faces were detected. If no faces are found, prints a message; otherwise, proceeds to draw rectangles around the detected faces.
- Draw Rectangles Around Faces:
- Display the Result:

In [61]:
```
img = cv2.imread('Jeffrey Hinton.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(gray)
if len(faces) == 0:
    print("Face not found")
else:

    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 255), 3) #draw rectangles around the faces with width =3,color
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face Detection')
    plt.axis('off')
    plt.show()
```

Experimenting with scale factor aand minimum Neighbors parameter of

- **Scale Factor (scaleFactor):** It compensates for faces appearing smaller the farther they are from the camera. A higher scale factor makes the algorithm more sensitive to detecting faces but may also increase false positives. A common value is 1.3.
- **Minimum Neighbors (minNeighbors):** It specifies how many neighbors each candidate rectangle should have to retain it. Higher values result in fewer detections but with higher confidence. Lower values may lead to more detections but with lower confidence. A common value is 5.

In [4]:
```
img = cv2.imread('multiple_faces.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(img, scaleFactor=1.3, minNeighbors=3)
if len(faces) == 0:
    print("Face not found")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 3)

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Face Detection')
plt.axis('off')
plt.show()
```

minNeighbors=9

In [5]:
```
img = cv2.imread('multiple_faces.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(img, scaleFactor=1.3, minNeighbors=9)
if len(faces) == 0:
    print("Face not found")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 3)

    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face Detection')
    plt.axis('off')
    plt.show()
```

scaleFactor=1.9, minNeighbors=9

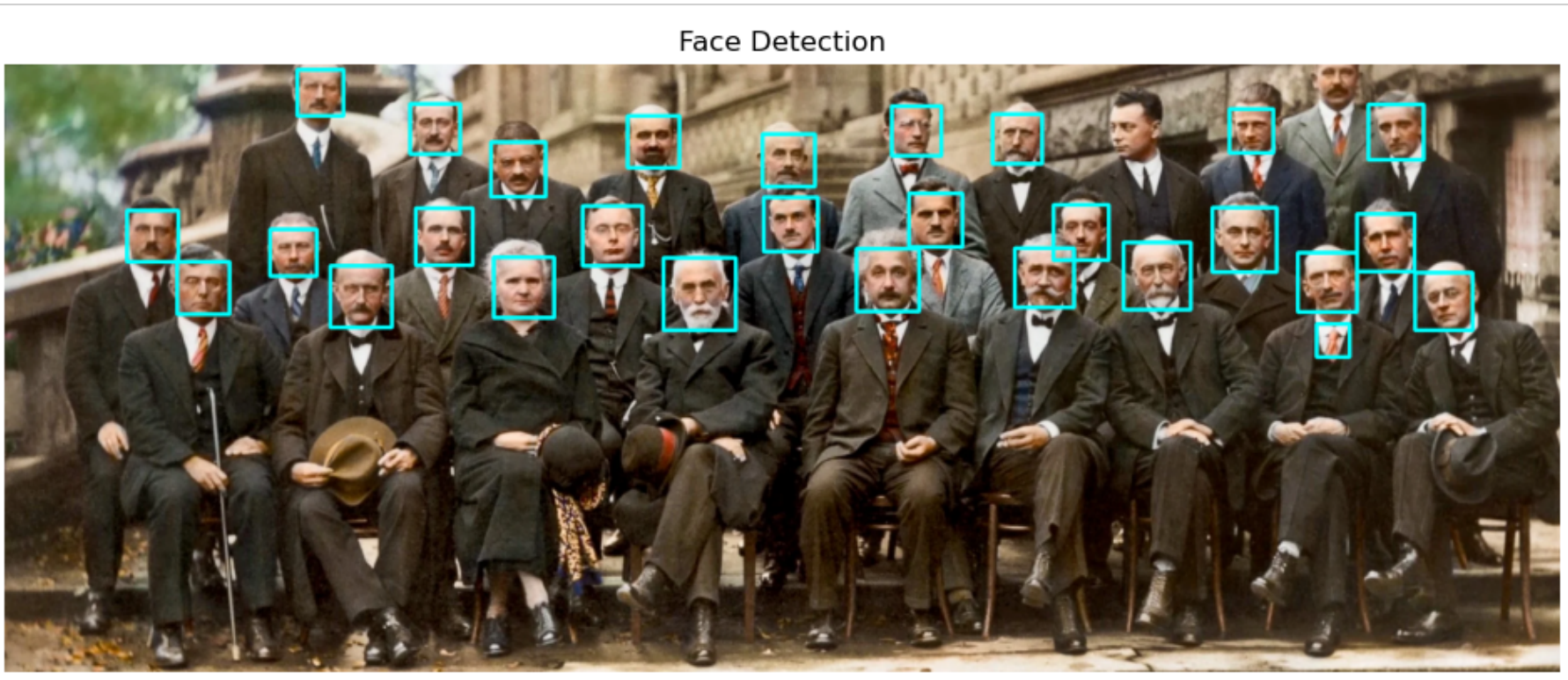
In [6]:
```
img = cv2.imread('multiple_faces.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(img, scaleFactor=1.9, minNeighbors=9)
if len(faces) == 0:
    print("Face not found")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 3)

    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face Detection')
    plt.axis('off')
    plt.show()
```

In [7]:
```
img = cv2.imread('multiple_faces.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(img, scaleFactor=2, minNeighbors=9)
if len(faces) == 0:
    print("Face not found")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 3)

    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face Detection')
    plt.axis('off')
    plt.show()
```

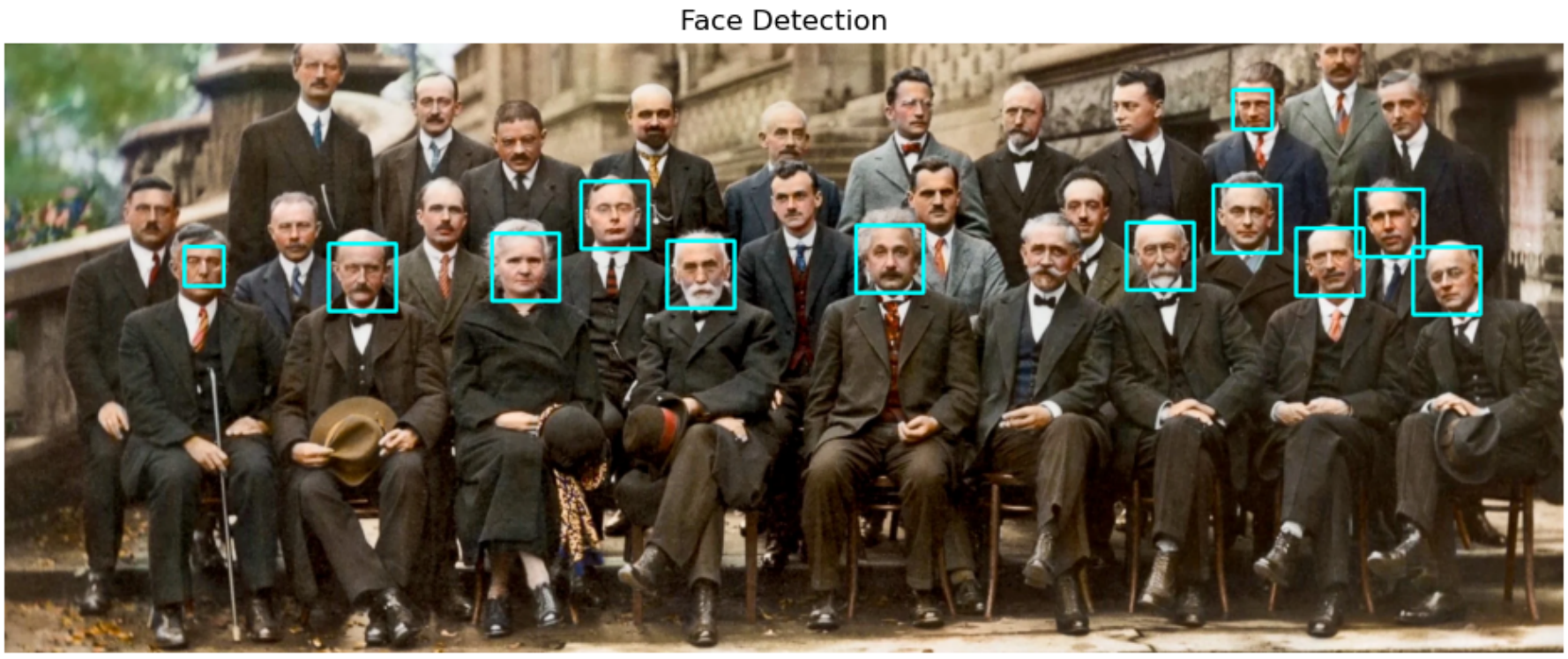
In [8]:
```
img = cv2.imread('multiple_faces.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(img, scaleFactor=1.4, minNeighbors=4)
if len(faces) == 0:
    print("Face not found")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 3)

    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face Detection')
    plt.axis('off')
    plt.show()
```

In [16]:
```
img = cv2.imread('girl_lamb.jpg',-1)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_classifier.detectMultiScale(gray, 1.3, 5)
if len(faces) == 0:
    print("****Face not found****")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 4)

    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face Detection')
    plt.axis('off')
    plt.show()
```

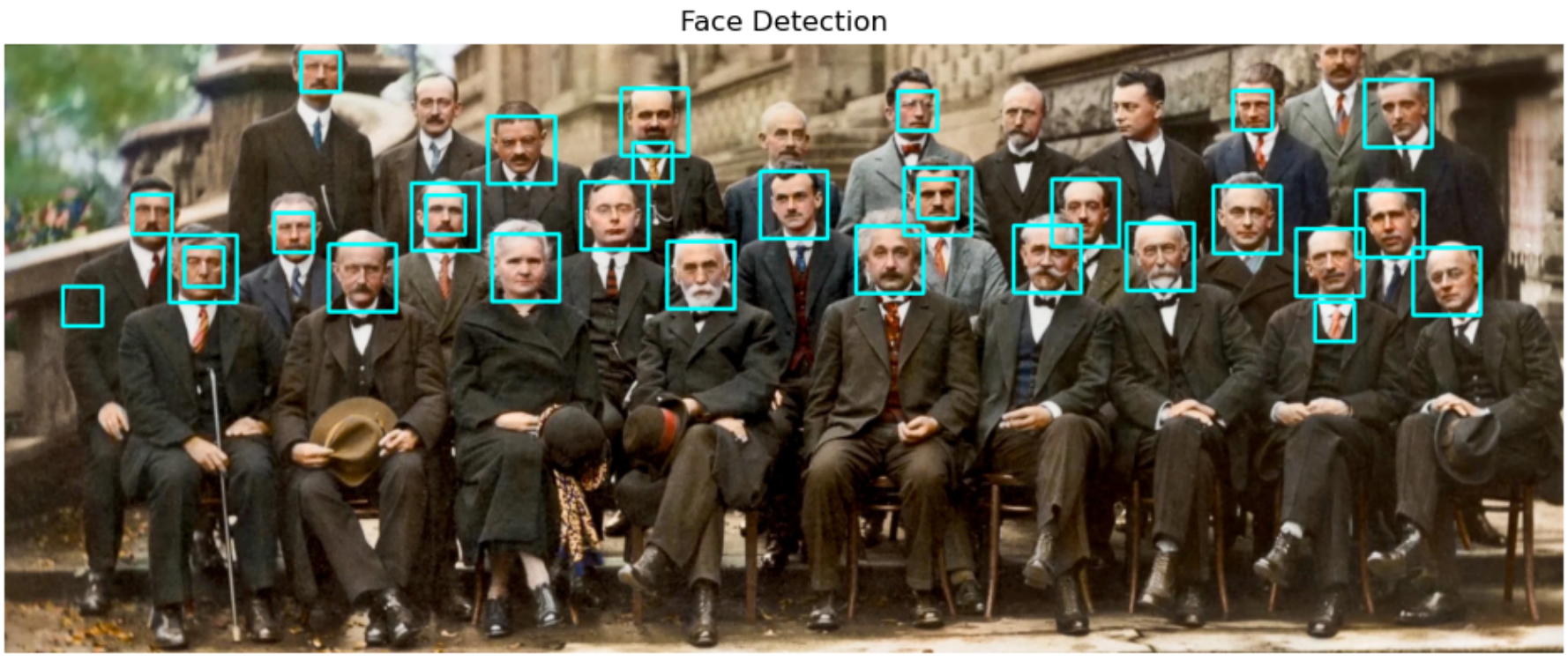
In [32]:
```
img = cv2.imread('scientists.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_classifier.detectMultiScale(gray, 1.3, 6)
if len(faces) == 0:
    print("****Face not found****")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 6)
    plt.figure(figsize=(12,8))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face Detection')
    plt.axis('off')
    plt.show()
```



In [29]:
```
img = cv2.imread('scientists.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_classifier.detectMultiScale(gray, 1.7, 6)
if len(faces) == 0:
    print("****Face not found****")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 6)
    plt.figure(figsize=(12,8))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face Detection')
    plt.axis('off')
    plt.show()
```

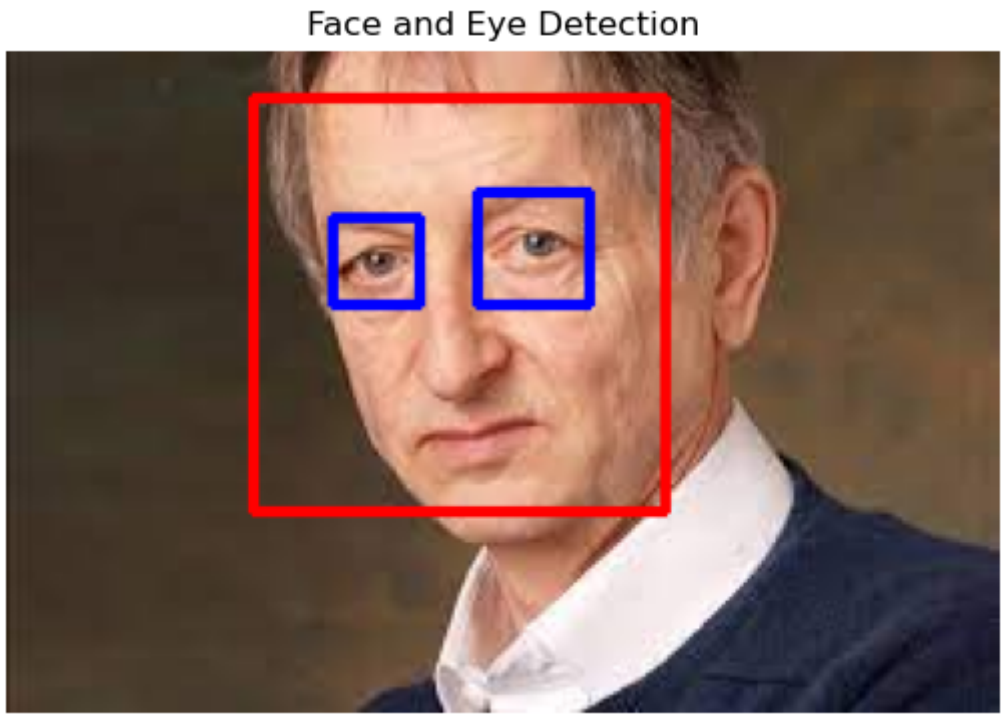


In [31]:
```
img = cv2.imread('scientists.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_classifier.detectMultiScale(gray, 1.7, 2)
if len(faces) == 0:
    print("****Face not found****")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 6)
    plt.figure(figsize=(12,8))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face Detection')
    plt.axis('off')
    plt.show()
```

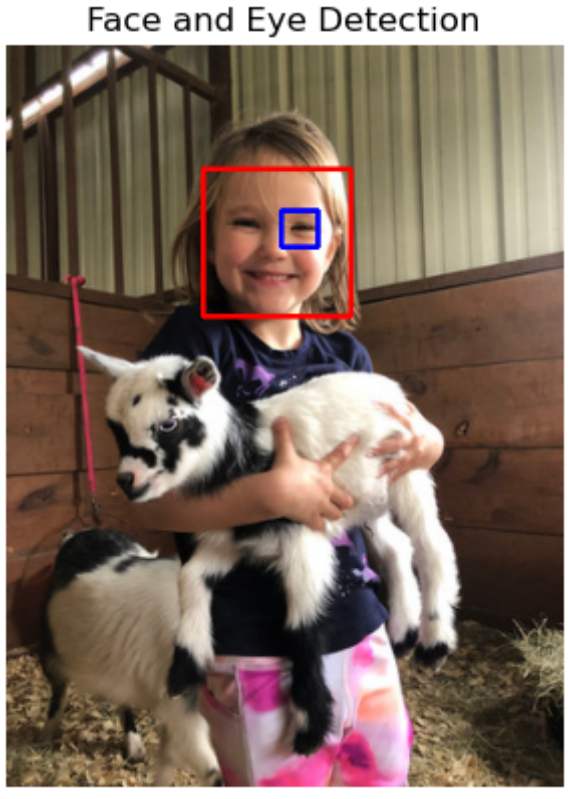


Eye Detection

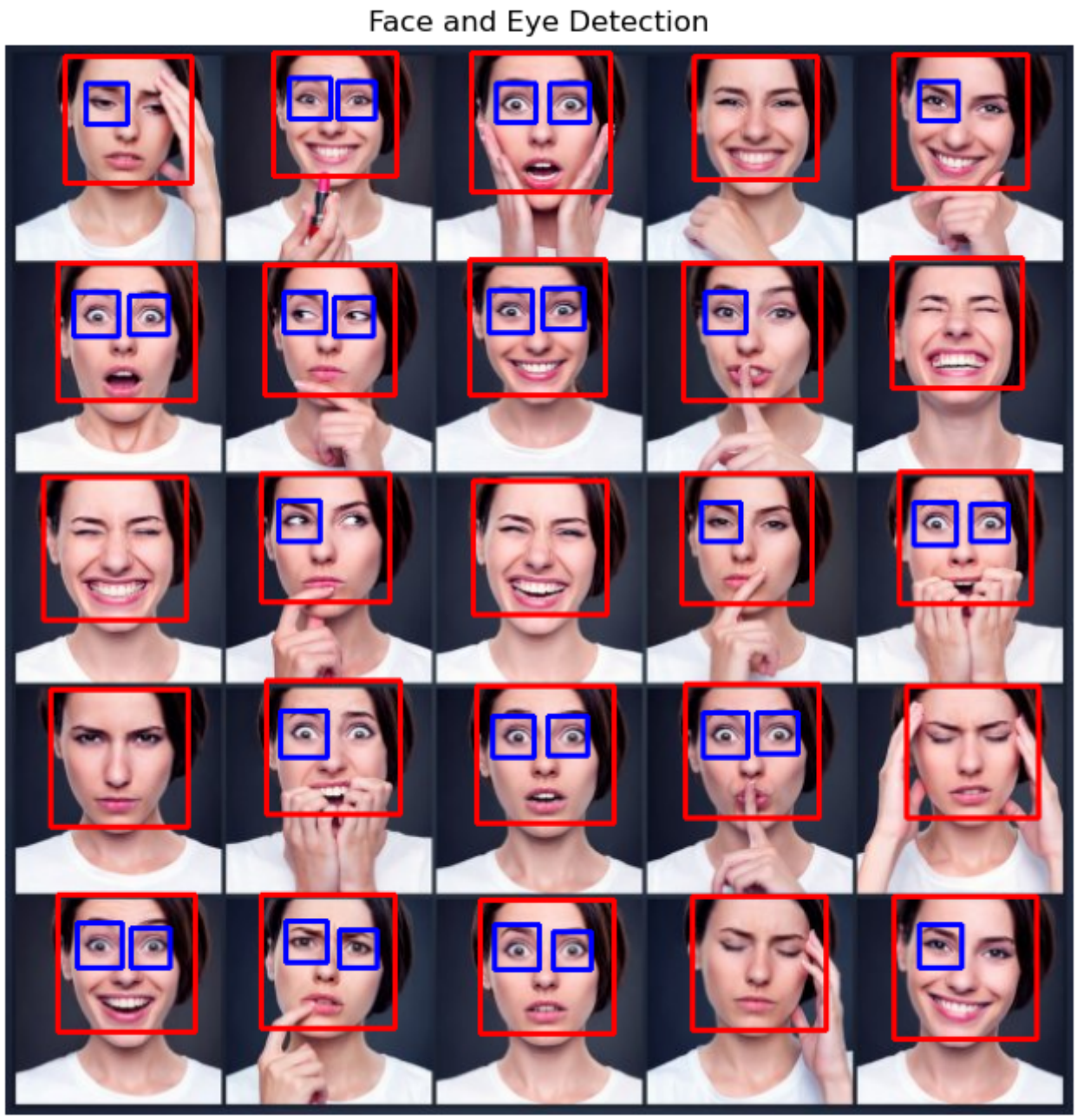
In [38]:
```
img = cv2.imread('Jeffrey Hinton.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(img, 1.3, 5)

if len(faces) == 0:
    print("Face not found")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)
        eye_color = img[y:y+h, x:x+w]

        eyes = eye_classifier.detectMultiScale(eye_color)
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(eye_color, (ex, ey), (ex+ew, ey+eh), (255, 0, 0), 2)


    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face and Eye Detection')
    plt.axis('off')
    plt.show()
```



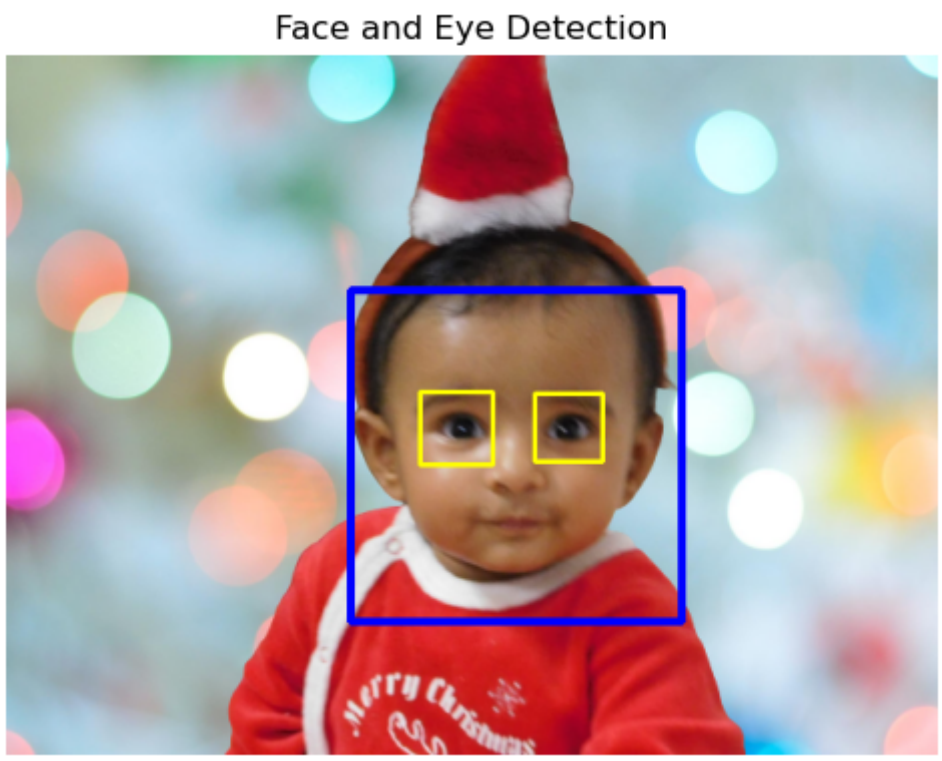
In [39]:
```
img = cv2.imread('girl_lamb.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(img, 1.3, 5)
# Face check
if len(faces) == 0:
    print("Face not found")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)
        eye_color = img[y:y+h, x:x+w]

        eyes = eye_classifier.detectMultiScale(eye_color)
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(eye_color, (ex, ey), (ex+ew, ey+eh), (255, 0, 0), 2)


    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face and Eye Detection')
    plt.axis('off')
    plt.show()
```



In [64]:
```
img = cv2.imread('expressions.jpeg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(img, 1.3, 5)
# Face check
if len(faces) == 0:
    print("Face not found")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)
        eye_color = img[y:y+h, x:x+w]

        eyes = eye_classifier.detectMultiScale(eye_color, 1.3,5)
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(eye_color, (ex, ey), (ex+ew, ey+eh), (255, 0, 0), 2)

    plt.figure(figsize =(8,8))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face and Eye Detection')
    plt.axis('off')
    plt.show()
```



In [74]:
```
img = cv2.imread('my_baby.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(img, 1.3, 5)
# Face check
if len(faces) == 0:
    print("Face not found")
else:
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 3)
        eye_color = img[y:y+h, x:x+w]

        eyes = eye_classifier.detectMultiScale(eye_color, 1.3,5)
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(eye_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 255), 2)

    plt.figure(figsize =(6,6))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Face and Eye Detection')
    plt.axis('off')
    plt.show()
```



Thanks for your attention.