1. Kullanılan Kütüphanelerin Dahil Edilmesi:

```
import cv2
import argparse
import numpy as np
```

Bu kısımda, görüntü işleme için OpenCV (`cv2`), komut satırı argümanlarını işlemek için `argparse` ve diziler ve matrisler üzerinde işlem yapmak için `numpy` kütüphaneleri dahil edilir.

2. Komut Satırı Argümanlarının İşlenmesi:

```
ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required=True,
                help='D:/Apple-Detection/apple_tree_images')
ap.add_argument('-c', '--config', required=True,
                help='D:/Apple-Detection/yolov3.cfg')
ap.add_argument('-w', '--weights', required=True,
                help='D:/Apple-Detection/yolov3.weights')
ap.add_argument('-cl', '--classes', required=True,
                help='D:/Apple-Detection/yolov3.txt')
```

Bu kısımda `argparse` kullanılarak komut satırından dört adet argüman alınır: görüntü dosyasının yolu (`--image`), YOLOv3 ağına ait yapılandırma dosyasının yolu (`--config`), ağırlıklar dosyasının yolu (`--weights`) ve sınıf etiketlerinin bulunduğu dosyanın yolu (`--classes`).

3. YOLO Modelinden Çıktı Katmanlarının İsimlerinin Alınması:

```
def get_output_layers(net):
    layer_names = net.getUnconnectedOutLayersNames()
    return layer_names
```

Bu işlev, YOLOv3 ağının çıkış katmanlarının isimlerini almak için kullanılır.

4. Algılanan Nesneleri Çizmek İçin İşlev:

```
def draw_prediction(img, class_id, confidence, x, y, x_plus_w, y_plus_h):
    label = str(classes[class_id])
    color = COLORS[class_id]
    cv2.rectangle(img, (x, y), (x_plus_w, y_plus_h), color, 2)
    cv2.putText(img, label, (x - 4, y - 4), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                color, 2)
```

Bu işlev, algılanan nesnelerin çizilmesi için kullanılır. Etiketin, algılanan sınıfın adını dikdörtgenin sol üst köşesine ve dikdörtgenin dışına yazılmasını sağlar.

5. Görüntünün Yüklenmesi:

```
image = cv2.imread(args.image)
if image is None:
    print("Görüntü yüklenirken hata oluştu.")
    exit()
```

Belirtilen görüntü dosyası yüklenir. Eğer görüntü yüklenemezse, bir hata mesajı yazdırılır ve program sonlandırılır.

6. Görüntü Özelliklerinin Alınması:

```
Width = image.shape[1]
Height = image.shape[0]
scale = 0.00392
classes = None
```

Bu satırlarda, görüntünün genişliği (`Width`) ve yüksekliği (`Height`) alınır. Ayrıca, görüntünün boyutlarını ölçeklemek için kullanılacak bir ölçek faktörü (`scale`) belirlenir.

7. Sınıf İsimlerinin Yüklenmesi:

```
with open(args.classes, 'r') as f:
    classes = [line.strip() for line in f.readlines()]

# Rastgele renkler oluştur
```

Bu kısımda, sınıf isimleri dosyadan (`args. Classes`) okunur ve bir liste olarak `classes` değişkenine atanır.

8. Rastgele Renklerin Oluşturulması:

```
COLORS = np.random.uniform(0, 255, size=(len(classes), 3))
```

Her bir sınıf için rastgele bir renk oluşturulur ve bu renkler `COLORS` dizisinde saklanır.

9. YOLOv3 Ağının Yüklenmesi:

```
net = cv2.dnn.readNet(args.weights, args.config)
```

Bu satırda, YOLOv3 modeli `cv2.dnn.readNet()` fonksiyonu kullanılarak ağırlıklar ve yapılandırma dosyalarından (`args.weights` ve `args.config`) yüklenir.

10. Görüntüden Blob Oluşturulması:

```
blob = cv2.dnn.blobFromImage(image, scale, (416, 416), (0, 0, 0), True,
                             crop=False)
```

Görüntü, YOLOv3 modeline uygun bir giriş blob'una dönüştürülür. Bu işlem, ağırlıkların işlenmesi için görüntünün ön işleme adımını oluşturur.

11. Ağın Çıkışlarının Alınması:

```
outs = net.forward(get_output_layers(net))
```

Ağa giriş olarak blob verilir ve çıkışlar alınır.

12. Nesne Tespitinin Gerçekleştirilmesi:

```
class_ids = []
confidences = []
boxes = []
conf_threshold = 0.4
nms_threshold = 0.3
apple_count = 0

for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > conf_threshold:
            center_x = int(detection[0] * Width)
            center_y = int(detection[1] * Height)
            w = int(detection[2] * Width)
            h = int(detection[3] * Height)
            x = center_x - w // 2
            y = center_y - h // 2
            class_ids.append(class_id)
            confidences.append(float(confidence))
            boxes.append([x, y, w, h])
```

Algılanan nesnelerin koordinatları, güven skorları ve sınıf kimlikleri `class_ids`, `confidences` ve `boxes` listelerine eklenir.

13. Non-Maximum Suppression (NMS) Uygulanması:

```
indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)
```

Algoritmik olarak, NMS, aynı nesneyi birden fazla kez algılayan kutular arasından en uygun olanını seçer.

14. Algılanan Nesnelerin Çizilmesi ve Sayılması:

```
if len(indices) > 0:
    for i in indices:
        box = boxes[i]
        x, y, w, h = box[0], box[1], box[2], box[3]
        draw_prediction(image, class_ids[i], confidences[i], round(x), round(y),
                        round(x + w), round(y + h))
        apple_count += 1
else:
    print("Fotoğrafta Elma tespit edilmedi.")
```

Algılanan nesnelerin çizilmesi ve sayılması işlemleri gerçekleştirilir.

15. Sonuçların Gösterilmesi ve Kaydedilmesi:

```
print(f"Fotoğraftaki Toplam Elma Sayısı: {apple_count}")


cv2.imshow("Elma Tespiti Penceresi", image)
cv2.waitKey()

cv2.imwrite("result.jpg", image)
cv2.destroyAllWindows()
```

Elde edilen sonuçlar ekranda gösterilir ve görüntü kaydedilir. Bu kısımlar, algılama işlemi tamamlandıktan sonra sonuçları göstermek ve işlemi sonlandırmak için kullanılır.