1. Kullanılan Kütüphanelerin Dahil Edilmesi:

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

Bu satırlarda, 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('-v', '--video', required=True,
                help='D:/Apple_Detection/apple_tree_images_and_vids/apple_garden.mp4')
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')
args = ap.parse_args()
```

Bu kısımda `argparse` kullanılarak komut satırından dört adet argüman alınır: video dosyasının yolu (`--video`), 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):
    # Algılanan nesnenin sınıf adını alır
    label = str(classes[class_id])
    # Algılanan nesnenin rengini belirler
    color = COLORS[class_id]

    # Algılanan nesnenin etrafına çerçeve çizer
    cv2.rectangle(img, (x, y), (x_plus_w, y_plus_h), color, 2)
    # Dikdörtgenin içine sınıf adını ve güven skorunu yazar
    cv2.putText(img, label + " " + str(round(confidence, 2)), (x - 10, y - 10),
```

Bu işlev, algılanan nesnelerin çizilmesi için kullanılır. Etiketin, algılanan sınıfın adını, güven skorunu, renkli bir dikdörtgen içine ve dikdörtgenin dışına yazılmasını sağlar.

5. Sınıf İsimlerinin Yüklenmesi:

```
classes = None
with open(args.classes, 'r') as f:
    classes = [line.strip() for line in f.readlines()]
```

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

6. Rastgele Renklerin Oluşturulması:

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

# Sinir ağını yükle
```

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

7. YOLOv3 Ağının Yüklenmesi:

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

# Videoyu yükle
```

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

8. Video Dosyasının Açılması ve İşlenmesi:

```
video = cv2.VideoCapture(args.video)
```

Video dosyası açılır ve işlenecek olan video nesnesi oluşturulur.

9. Video Üzerinde YOLO Modeliyle Nesne Tespiti:

```python
while True:
    # Videodan bir kare al ve eğer kare alınamazsa döngüyü kır
    ret, frame = video.read()
    if not ret:
        break

    # Kare genişliğini ve yüksekliğini al
    Width = frame.shape[1]
    Height = frame.shape[0]
    scale = 0.00392

    # YOLO modeline uygun blob oluştur
    blob = cv2.dnn.blobFromImage(frame, scale, (416, 416), (0, 0, 0), True, crop=False)
    net.setInput(blob)

    # YOLO ile tespit yap
    outs = net.forward(get_output_layers(net))

    # Değişken ve dizi tanımlamaları
    class_ids = []
    confidences = []
    boxes = []
    conf_threshold = 0.4
    nms_threshold = 0.3

    # YOLO modelinin çıktılarını işleyerek tespit edilen nesnelerin bilgilerini topla
    # Algılanan nesneler için güven skoru belirli bir eşik değerinden büyükse, sınıf ID
    # Sonuç olarak, tespit edilen nesnelerin sınıf ID'leri, güven skorları ve sınırlayı
    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])
```

Bu kısımda, her bir video karesi üzerinde YOLO modeli kullanılarak nesne tespiti yapılır. Algılanan nesnelerin sınıf ID'leri, güven skorları ve sınırlayıcı kutuları (`boxes`) listelenir.

10. Non-Maximum Suppression (NMS) Uygulanması ve Algılanan Nesnelerin Çizilmesi:

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


    # Algılanan nesneleri çerçevelerle işaretleme
    if len(indices) > 0:
        for i in indices:
            kutu = boxes[i]
            x, y, w, h = kutu[0], kutu[1], kutu[2], kutu[3]
            draw_prediction(frame, class_ids[i], confidences[i], round(x),
                            round(y), round(x + w), round(y + h))
    else:
        print("Elma algılanmadı.")


    # Video kodunu çalıştırdıktan sonra esc tuşuna basarak durdur
    cv2.imshow("Video Penceresi", frame)
    if cv2.waitKey(1) == 27:
        break
```

Algılanan nesnelerin çizilmesi ve eğer elma algılanmazsa bir mesaj yazdırılması işlemleri gerçekleştirilir. Bu işlemler video üzerinde her kare için tekrarlanır.

11. Video ve Pencerelerin Kapatılması:

```
# Video yakalama ve pencereleri kapat
video.release()
cv2.destroyAllWindows()
```

Video dosyasının serbest bırakılması ve tüm pencerelerin kapatılması işlemleri gerçekleştirilir.