EXECUTIONAL PROCESS

Step 1: Set Up Environment First, ensure you have Python installed on your computer. Then, install the necessary libraries:

```
pip install opencv-python
pip install opencv-python-headless
pip install numpy
pip install imutils
pip install dlib
```

Step 2: Load YOLO Model for Car Detection For object detection, we'll use the YOLO (You Only Look Once) algorithm with pre-trained weights. Download the YOLO model files (yolov3.weights and yolov3.cfg) from the official website or use YOLOv4 or YOLOv5 if you prefer.

```
import cv2
import numpy as np


# Load YOLO model
net = cv2.dnn.readNet("path/to/yolov3.weights", "path/to/yolov3.cfg")
classes = []
with open("path/to/coco.names", "r") as f:
    classes = f.read().strip().split("\n")
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

Step 3: Car Detection and Tracking Next, we'll create a function to detect and track cars in the video feed using the YOLO model

```
def detect_cars(frame):
    height, width = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

    net.setInput(blob)
    outs = net.forward(output_layers)

    class_ids = []
    confidences = []
    boxes = []

    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5 and class_id == 2:  # 2 represents cars class in COCO dataset
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
```

```
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

    indices = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    cars = []
    for i in indices:
        i = i[0]
        car = boxes[i]
        cars.append(car)

    return cars
```

Step 4: Object Tracking using CentroidTracker

Now, we'll implement object tracking to track cars across frames using the CentroidTracker algorithm.

```
from imutils.video import VideoStream
from centroid_tracker import CentroidTracker

# Initialize CentroidTracker
ct = CentroidTracker(max_disappeared=50, max_distance=50)

# Initialize video stream
vs = VideoStream(src=0).start()
```

```
while True:
    frame = vs.read()
    if frame is None:
        break

    # Detect cars in the frame
    cars = detect_cars(frame)

    # Update CentroidTracker with new detections
    objects = ct.update(cars)

    for (objectID, centroid) in objects.items():
        # Draw the car's ID and centroid on the frame
        text = f"Car {objectID}"
        cv2.putText(frame, text, (centroid[0] - 10, centroid[1] - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
        cv2.circle(frame, (centroid[0], centroid[1]), 4, (0, 255, 0), -1)

    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    if key == ord("q"):
        break

cv2.destroyAllWindows()
vs.stop()
```

Step 5: Parking Assistance and Control

Now, you can implement the parking assistance and control algorithm to guide the car to the parking spot. This part will depend on your hardware setup and how you want the car to move.

Step 6: Testing and Refinement

Test the system thoroughly, and make any necessary adjustments to improve performance and accuracy.