Parking spot detection and counter

This project is a computer vision system that detects which parking spaces are free or occupied from a video stream and keeps a live count of available spots using OpenCV and a pretrained classifier model.

Project overview

The Parking Space Counter takes a parking-lot video and a binary mask image marking each parking spot, then decides for every frame whether each spot is empty or occupied and overlays green/red boxes plus a free-spot counter. The core logic is implemented in main.py, which uses helper functions from util.py and a pretrained model file (model.p) .

Objectives

- Detect all defined parking spots in each video frame using a manually prepared mask image.
- Classify each spot as “empty” or “not empty” using a small image classifier model and update a real-time counter of free spaces.
- Optimize performance by avoiding unnecessary reclassification when a spot’s appearance has not significantly changed between frames.

System architecture

- **Input data**: A parking-lot video (video.avi or similar) and a mask image where each parking space is drawn as a white rectangle on a black background.
- **Processing pipeline**:
    - Read frames from the video using OpenCV.
    - Use the mask and utility functions to extract coordinates of each parking spot and crop the corresponding region from the current frame.
    - For selected frames or when sufficient pixel difference is detected compared to the previous frame, pass the crop to the classifier to predict empty vs occupied.
- **Outputs**: Colored rectangles (green for free, red for occupied) drawn on the video plus an on-screen text counter of total and free spots.

Key components

- **main.py**: Entry point that loads the video, mask image, and pretrained model, then runs the main processing loop and visualization.
- **util.py**: Contains helper functions to load spot coordinates from the mask, crop parking spots from frames, compute frame differences, and manage state between iterations.
- **Model and data folder (from Drive)**: Includes the example parking video, mask images, and the trained classifier (model.p) that predicts whether a cropped spot contains a car.
- **requirements.txt**: Lists Python dependencies such as OpenCV and other libraries needed to run the project.

Implementation details

- The classifier is trained separately on cropped images of parking spots labeled as empty or not empty, and the resulting model is serialized to model.p for use in the main script.
- The system tracks pixel-intensity differences between the current and previous frame for each spot, and only re-runs the classifier if the difference exceeds a threshold, which keeps the application responsive even for many parking spots.
- The code uses standard OpenCV operations (frame reading, masking, cropping, drawing rectangles, text overlay) and can be adapted to other parking lots by updating the mask and retraining the model with new crops.

How to run

- Install the dependencies from requirements.txt in a Python environment.
- Download the provided video, mask, and model files from the linked Google Drive folder and place them in the project directory.
- Run main.py; the script will open a window showing the parking-lot video with colored boxes and the dynamic counter of available spaces.