**Project Overview:** The goal of this project is to develop a system that can count the number of available parking spaces in an image or a video feed using Python and the OpenCV library. The system will utilize image processing techniques to detect and analyze parking spaces, distinguishing between empty and occupied spots.

**Project Steps:**

1. **Data Collection:** Obtain the image or video feed of the parking area. This can be from a camera feed, a recorded video, or a series of images.
2. **Image Preprocessing:** Preprocess the images or frames from the video feed to enhance their quality and improve the accuracy of object detection. Common preprocessing steps include resizing, noise reduction, and contrast adjustment.
3. **Parking Space Detection:** Use object detection techniques to identify the parking spaces in the image. This might involve training a deep learning model (like YOLO or Faster R-CNN) on a dataset of parking space images, or using pre-trained models and fine-tuning them for your specific scenario.
4. **Occupancy Classification:** Once you've detected the parking spaces, apply image segmentation or classification techniques to determine if each parking space is empty or occupied. This might involve setting up a threshold for color or intensity to differentiate between empty and occupied spots.
5. **Counting Algorithm:** Develop an algorithm to count the number of empty and occupied parking spaces based on the results of the occupancy classification. You could track changes in occupancy over time to update the count dynamically.
6. **Display and Visualization:** Display the processed images or video frames with the parking spaces marked as either empty or occupied, along with the corresponding counts. This could be in the form of visual overlays on the original image or a separate visualization.
7. **Testing and Validation:** Test your system on different images or video feeds to ensure its accuracy and reliability. Adjust the parameters and fine-tune the algorithms as needed.
8. **Optimization:** If the system's performance is not satisfactory, you might need to optimize various aspects such as detection accuracy, processing speed, and memory usage.
9. **Deployment:** Once you're satisfied with the performance, deploy the system to process live video feeds or images from cameras in a real parking area.

**Resources:**

* OpenCV documentation and tutorials for image processing.
* Object detection frameworks like YOLO or Faster R-CNN.
* Python libraries for machine learning, such as TensorFlow or PyTorch.
* Online courses or tutorials on computer vision and deep learning.

Remember that this project involves multiple complex steps, especially if you're training your own object detection model. It's important to break down each step and understand the concepts thoroughly. Good luck with your "Parking Space Count Using Python OpenCV" project!