Car Counting Using Morphological Operations

May 11, 2025

Abstract

This document demonstrates a car counting system using morphological image processing techniques. The pipeline includes image preprocessing, binarization, dilation, and connected component analysis to count vehicles in a traffic scene.

1 Libraries Used

- `numpy`: For numerical operations and array manipulation
- `cv2` (OpenCV): For image processing and computer vision operations
- `matplotlib.pyplot`: For image visualization and plotting

2 Step-by-Step Process

2.1 Step 1: Import Libraries

```python
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
```

2.2 Step 2: Download Image

Download the sample traffic image:

```
!wget https://raw.githubusercontent.com/AsadiAhmad/Car-Counter-
    Morphology/main/Pictures/Cars.jpg -O Cars.jpg
```

2.3 Step 3: Load and Display Image

Load the image in grayscale and display:

```
cars = cv.imread("Cars.jpg", cv.IMREAD_GRAYSCALE)
plt.imshow(cars, cmap="gray")
```

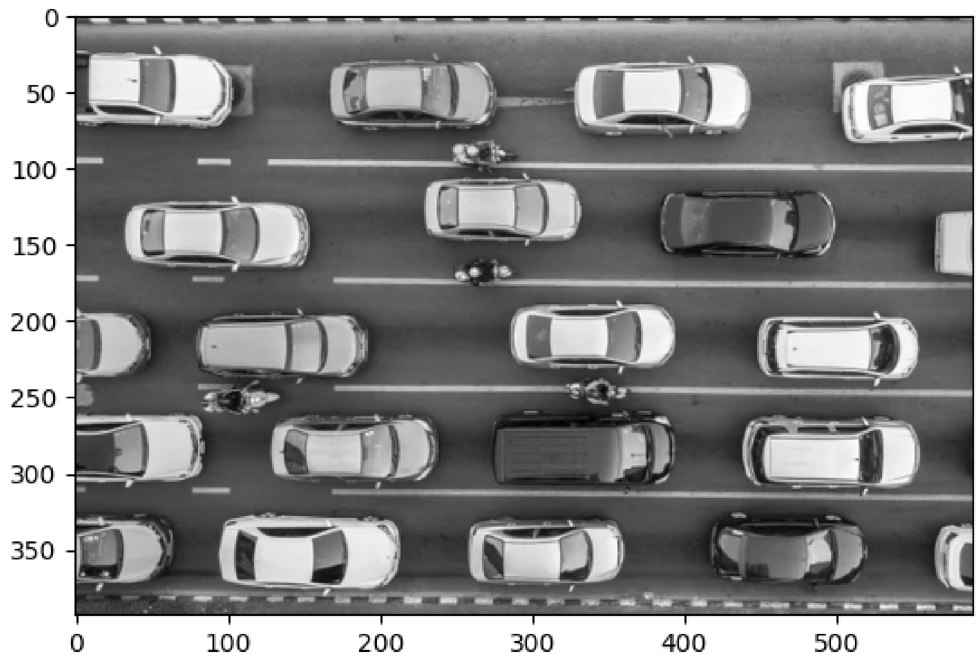

Figure 1: Original traffic image

2.4 Step 4: Crop Image

Remove unnecessary borders from the image:

```
height, width = cars.shape
cropped_cars = cars[15:height-15, :]
plt.imshow(cropped_cars, cmap="gray")
```

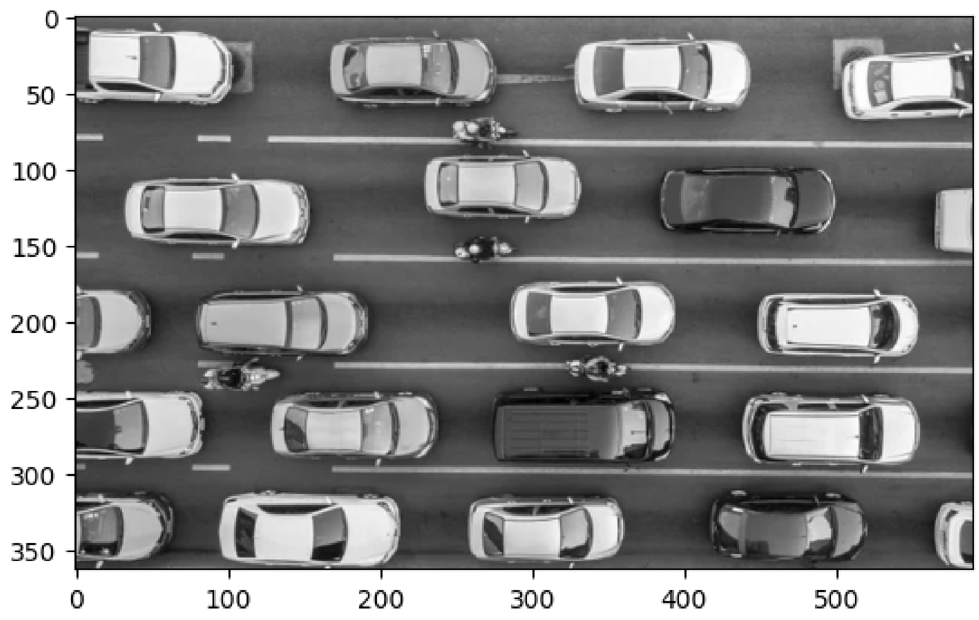

Figure 2: Cropped image removing top/bottom borders

2.5 Step 5: Blur Image

Apply Gaussian blur to reduce noise:

```
blurred_cars = cv.GaussianBlur(cropped_cars, (51, 51), 0)
plt.imshow(blurred_cars, cmap="gray")
```

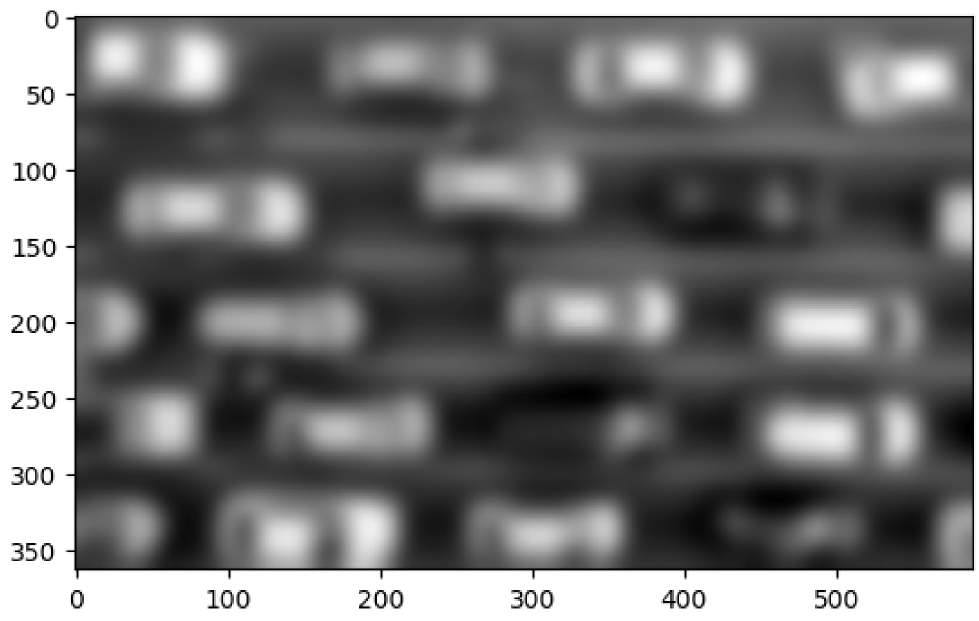

Figure 3: Image after Gaussian blur (51×51 kernel)

2.6 Step 6: Binarization

Convert to binary image for morphological processing:

```
bin_cars = np.where(blurred_cars > 127, 255, 0)
plt.imshow(bin_cars, cmap="gray")
scaled_cars = bin_cars.astype(np.uint8)
```

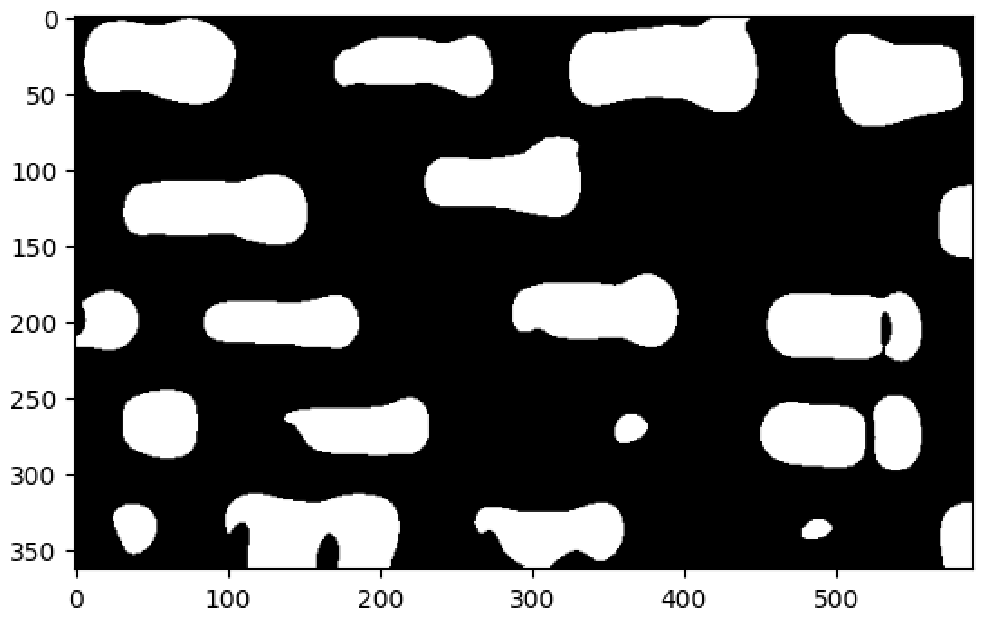

Figure 4: Binarized image (threshold = 127)

2.7 Step 7: Dilation for Connecting Car Parts

Connect car components using dilation:

```
kernel = np.ones((3, 3), np.uint8)
car_dilate = cv.dilate(scaled_cars, kernel, iterations=6)
plt.imshow(car_dilate, cmap="gray")
```

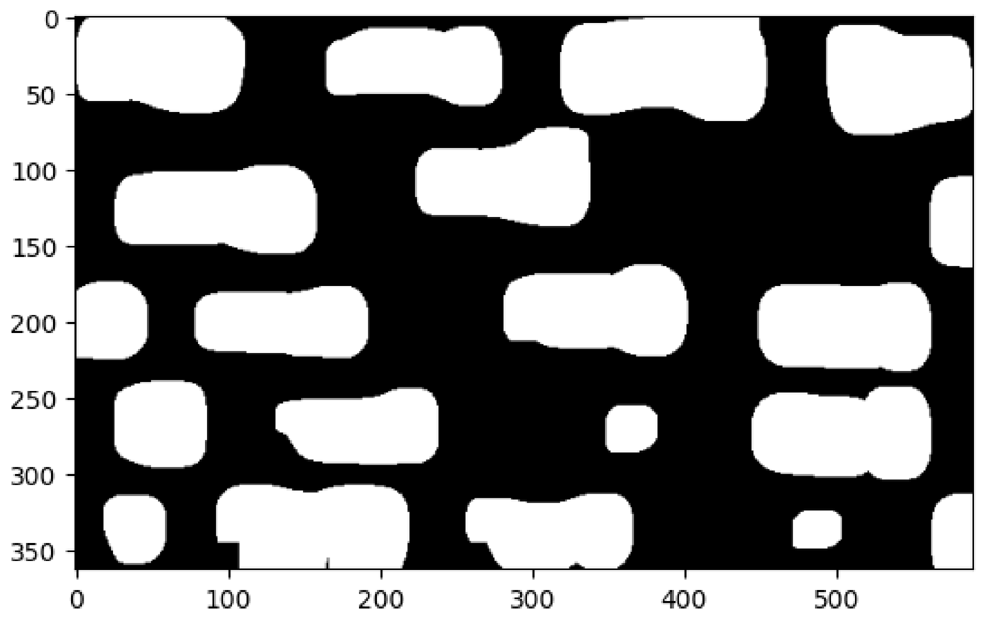

Figure 5: Image after dilation (6 iterations)

2.8 Step 8: Counting Cars

Count connected components:

```
num_labels, labels = cv.connectedComponents(car_dilate)
car_count = num_labels - 1  # Subtract background
print(car_count)
```

Output:

```
20
```

2.9 Step 9: Calculating Accuracy

Calculate counting accuracy:

```
percent = (car_count/21)*100
print(f'{percent:.2f}%')
```

Output:

```
95.24%
```

3 Technical Explanations

3.1 Morphological Processing

- **Cropping**: Removes irrelevant image regions

- **Gaussian Blur**: Reduces noise while preserving edges
- **Binarization**: Simplifies image for morphological operations
- **Dilation**: Connects nearby car parts into single components
- **Connected Components**: Identifies and counts individual cars

3.2 Performance Notes

- 95.24% accuracy achieved on sample image
- Kernel size and iterations affect counting results
- Works best with clear separation between vehicles

GitHub

`https://github.com/AsadiAhmad/Car-Counter-Morphology`