Sunflower Counting Using Color Segmentation and Morphology

May 11, 2025

Abstract

This document presents an automated sunflower counting system using color segmentation and morphological image processing. The pipeline includes HSV color filtering, noise removal, and connected component analysis to count sunflowers in an image.

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

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

2.2 Step 2: Download Image

Download the sunflower field image:

```
!wget https://raw.githubusercontent.com/AsadiAhmad/Sun-Flower-
   Counter/main/Pictures/sun_flower.jpg -O sun_flower.jpg
```

2.3 Step 3: Load and Display Image

Load and display the original image:

```python
sun_flower = cv.imread("sun_flower.jpg")
plt.imshow(sun_flower[...,::-1])  # Convert BGR to RGB for display
```

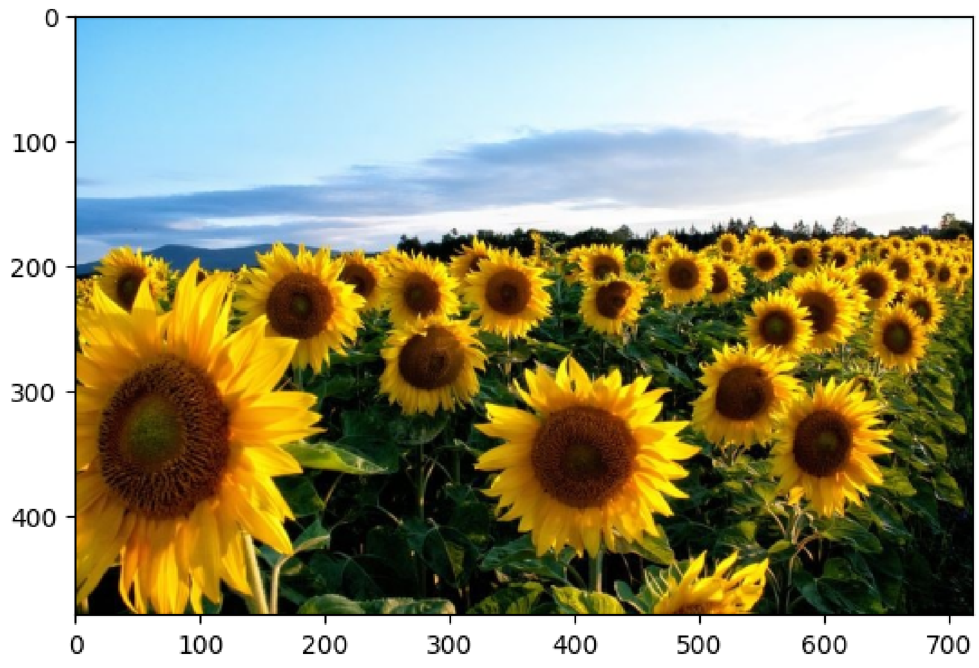

Figure 1: Original sunflower field image

2.4 Step 4: Convert to HSV Color Space

Transform to HSV for better color segmentation:

```python
hsv_sun_flower = cv.cvtColor(sun_flower, cv.COLOR_BGR2HSV)
```

2.5 Step 5: Filter Brown Sunflower Centers

Isolate brown sunflower centers using HSV range:

```python
lower_brown = np.array([0, 50, 0])    # Lower HSV bound
upper_brown = np.array([20, 255, 75]) # Upper HSV bound
masked_sun_flower = cv.inRange(hsv_sun_flower, lower_brown,
   upper_brown)
filtered_sun_flower = cv.bitwise_and(sun_flower, sun_flower, mask=
   masked_sun_flower)
```

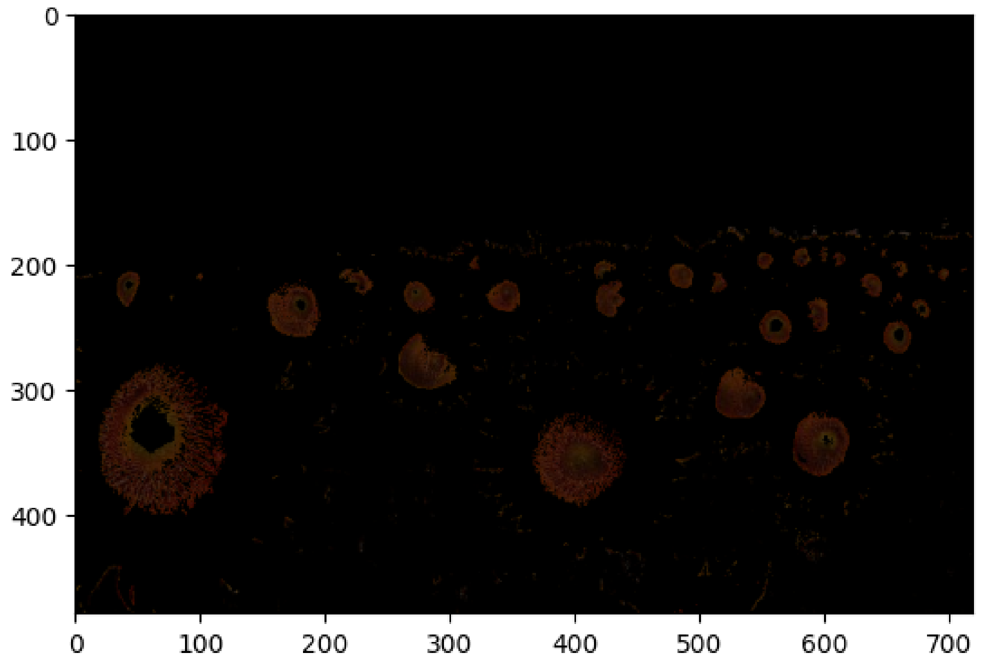

Figure 2: Image after brown color segmentation

2.6 Step 6-7: Convert to Grayscale and Binarize

Prepare for morphological processing:

```
filtered_sun_flower_gray = cv.cvtColor(filtered_sun_flower, cv.
    COLOR_BGR2GRAY)
filtered_sun_flower_bin = np.where(filtered_sun_flower_gray > 10,
    255, 0)
scaled_flower_bin = filtered_sun_flower_bin.astype(np.uint8)
```

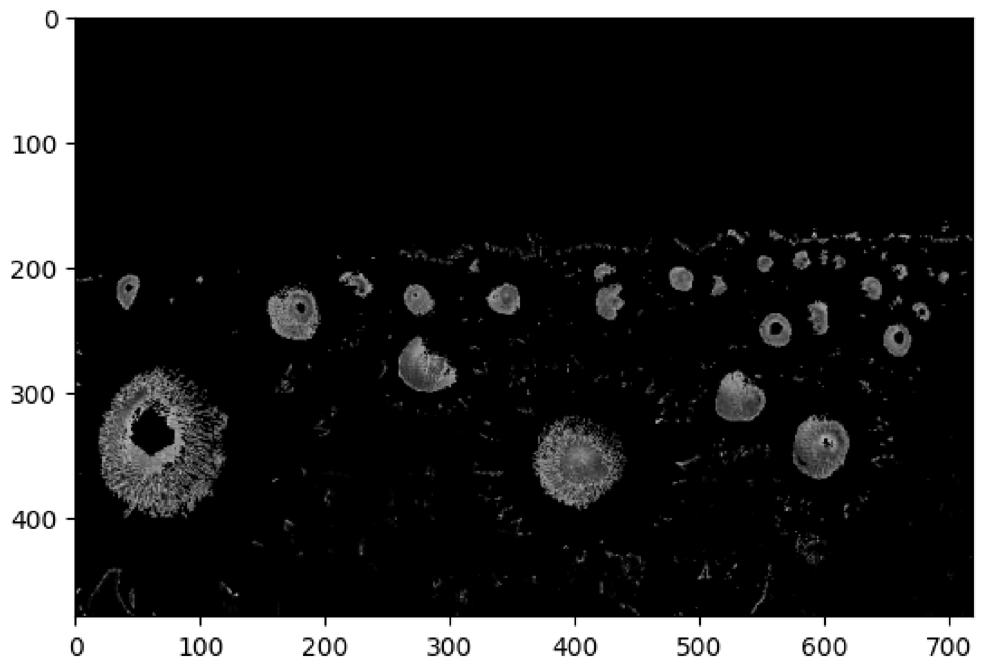

Figure 3: Binarized image of sunflower centers

2.7 Step 8-9: Morphological Erosion

Remove small noise artifacts:

```python
erosion_kernel = np.ones((2, 2), np.uint8)
erosion_sun_flower = cv.erode(scaled_flower_bin, erosion_kernel,
   iterations=3)
```

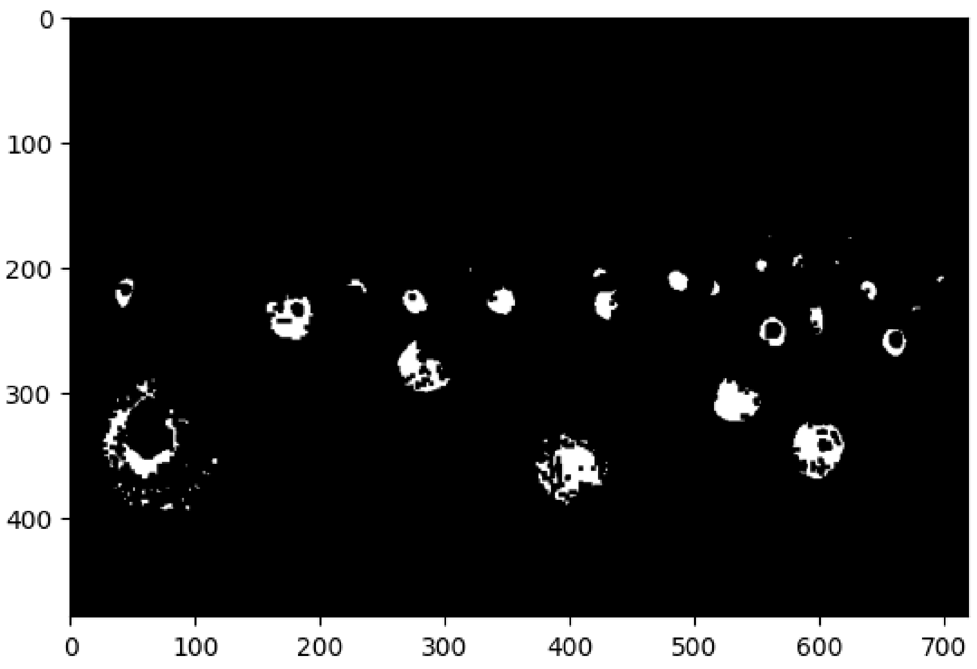

Figure 4: Image after erosion (noise removal)

2.8 Step 10: Morphological Dilation

Connect broken parts of sunflower centers:

```
dilation_kernel = np.ones((3, 3), np.uint8)
dilation_sun_flower = cv.dilate(erosion_sun_flower, dilation_kernel,
    iterations=5)
```

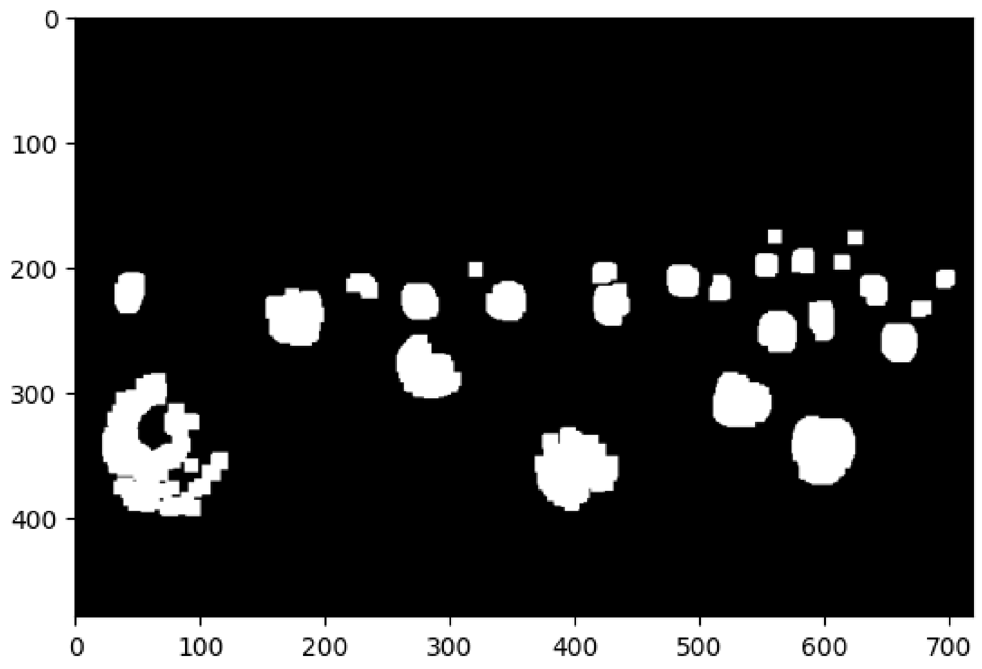

Figure 5: Image after dilation (connecting parts)

2.9 Step 11: Counting Sunflowers

Count connected components:

```python
num_labels, labels = cv.connectedComponents(dilation_sun_flower)
flower_count = num_labels - 1  # Subtract background
print(flower_count)
```

Output:

```
26
```

2.10 Step 12: Calculating Accuracy

Calculate counting accuracy:

```python
percent = (flower_count/29)*100
print(f'{percent:.2f}%')
```

Output:

```
89.66%
```

3 Technical Explanations

3.1 Image Processing Pipeline

- **HSV Color Space**: Better for color segmentation than RGB

- **Brown Color Range**: Carefully tuned to capture sunflower centers
- **Morphological Operations**: Erosion removes noise, dilation connects parts
- **Connected Components**: Counts individual sunflower centers

3.2 Performance Notes

- 89.66% accuracy achieved on sample image
- Color range selection critical for good segmentation
- Kernel sizes and iterations affect counting results

GitHub

`https://github.com/AsadiAhmad/Sun-Flower-Counter`