1) با استفاده از تکه کد اول ، تصویر را از روی فایل تکست میسازیم :

1. with open(file_path, 'r') as file:

فایل متنی (input1.txt) را برای خواندن باز می‌کند.

2. width, height, _ = map(int, lines[0].split())

اولین خط فایل را می‌خواند که شامل عرض و ارتفاع است؛

3. pixels = []

یک لیست خالی ایجاد می‌کند که برای ذخیره‌سازی داده‌های پیکسلی به‌صورت RGB استفاده خواهد شد.

4. for line in lines[1:]:

pixel_data = line.replace('(', '').replace(')', ',').replace(' n', '').split(',')

pixels.extend((int(pixel_data[i]), int(pixel_data[i+1]), int(pixel_data[i+2])) for i in range(0, len(pixel_data) - 1, 3))

- ابتدا هر پیکسل را از فرمت (R, G, B) به اعداد جداگانه تجزیه می‌کند.

- سپس مقادیر RGB هر پیکسل را به‌صورت تاپل های (سه‌تایی) عددی در لیست pixels ذخیره می‌کند.

5. image = Image.new("RGB", (height, width))

یک تصویر جدید با حالت رنگی RGB و ابعاد مشخص‌شده ایجاد می‌کند.

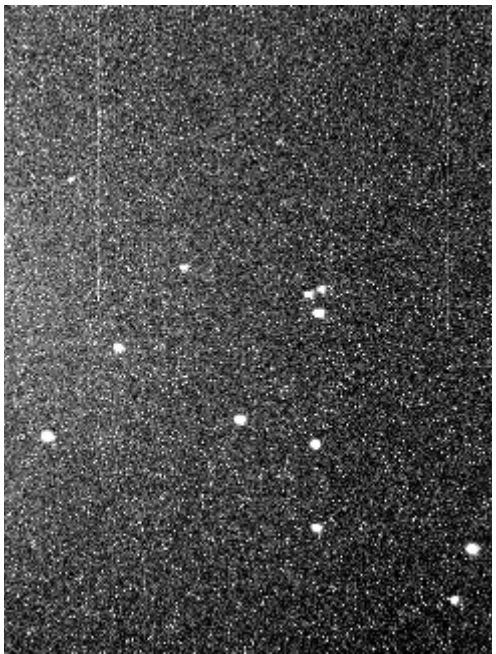
6. image.putdata(pixels)

داده‌های پیکسلی را به تصویر می‌افزاید، به طوری که هر تریپل در لیست pixels به یک پیکسل در تصویر تبدیل می‌شود.

7. image.save(output_path)

تصویر ساخته‌شده را با نام output_image.png ذخیره می‌کند.

خروجی:



2) تکه کد دوم ، ماسک ستارگان را ایجاد میکند و با استفاده از آن ، تصویری جدید فقط شامل ستارگان ایجاد میکند :

1. image = cv2.imread('output_image.png')

تصویر output_image.png را بارگذاری می‌کند که از قبل به‌صورت RGB ذخیره شده است.

2. gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

تصویر را به سطح خاکستری تبدیل می‌کند تا پردازش روی آن آسان‌تر باشد.

3. equalized = exposure.equalize_adapthist(gray, clip_limit=0.03)

کنتراست تصویر را افزایش می‌دهد.

4. equalized = (equalized * 255).astype(np.uint8)

مقدار نرمال شده تصویر را به مقیاس 0 تا 255 تبدیل کرده و به فرمت صحیح uint8 برای پردازش‌های بعدی تغییر می‌دهد.

5. thresholded = cv2.threshold(equalized, 150, 255, cv2.THRESH_BINARY) ,_

آستانه‌گذاری باینری را اعمال می‌کند؛ پیکسل‌هایی که مقدارشان بیشتر از 150 است، به 255 (سفید) و بقیه به 0 (سیاه) تبدیل می‌شوند.

6. kernel = np.ones((3, 3), np.uint8) و cleaned = cv2.morphologyEx(thresholded, cv2.MORPH_OPEN, kernel)

نویز تصویر را با استفاده از عملگر مورفولوژیک باز (opening) حذف می‌کند. این عمل از یک کرنل 3x3 برای حذف نویزهای کوچک در تصویر استفاده می‌کند.

7. contours, _ = cv2.findContours(cleaned, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

کانتورهای نواحی سفید در تصویر را شناسایی می‌کند.

8. mask = np.zeros_like(cleaned)

یک ماسک خالی به اندازه تصویر ساخته می‌شود تا نواحی مورد نظر در آن رسم شوند.

9. for contour in contours:

if cv2.contourArea(contour) >= min_star_size:

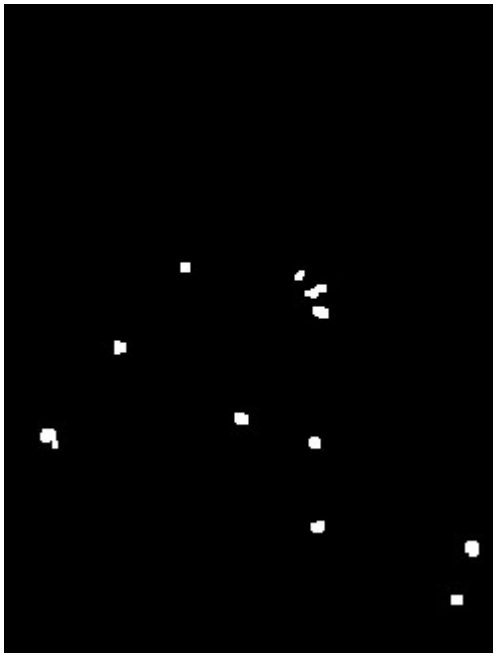
cv2.drawContours(mask, [contour], -1, 255, thickness=cv2.FILLED)

در این حلقه، کانتورهایی با مساحت بزرگتر یا مساوی min_star_size (10 پیکسل) انتخاب می‌شوند و به‌صورت سفید (255) روی ماسک رسم می‌شوند. این کار نواحی کوچک را فیلتر می‌کند.

10. cv2.imwrite('filtered_stars_mask.jpg', mask)

ماسک نهایی را به نام filtered_stars_mask.jpg ذخیره می‌کند که فقط نواحی مورد نظر (نواحی با سایز مناسب) را شامل می‌شود.

خروجی :



3) تکه کد سوم برای شمارش و گزارش مختصات ستارگان :

1. mask = cv2.imread('filtered_stars_mask.jpg', cv2.IMREAD_GRAYSCALE)

تصویر ماسک filtered_stars_mask.jpg را به صورت خاکستری بارگذاری می‌کند که شامل نواحی شناسایی‌شده است.

2. _, binary_mask = cv2.threshold(mask, 127, 255, cv2.THRESH_BINARY)

آستانه‌گذاری باینری را روی ماسک انجام می‌دهد تا نواحی سفید (ستاره‌ها) به 255 (سفید) و نواحی سیاه به 0 (سیاه) تبدیل شوند.

3. labeled_array, num_features = label(binary_mask)

با استفاده از تابع label از scipy.ndimage ، نواحی متصل (مناطق ستاره‌ها) در تصویر را شناسایی کرده و به هر ناحیه یک برچسب (label) منحصر به فرد اختصاص می‌دهد. همچنین تعداد ویژگی‌ها (همان ستاره‌ها) را برمی‌گرداند.

4. print("Number of stars detected:", num_features)

تعداد ستاره‌های شناسایی‌شده را چاپ می‌کند.

5. star_centers = []

یک لیست خالی برای ذخیره مختصات مراکز ستاره‌ها ایجاد می‌کند.

6. for i in range(1, num_features + 1):

slices = find_objects(labeled_array == i)

برای هر برچسب (ستاره) از 1 تا num_features ، نواحی مربوطه را پیدا می‌کند.

7. if slices:

y_slice, x_slice = slices[0]

cX = int((x_slice.start + x_slice.stop - 1) / 2)

cY = int((y_slice.start + y_slice.stop - 1) / 2)

star_centers.append((cX, cY))

- اگر ناحیه‌ای پیدا شد:

- محدوده‌های افقی و عمودی آن ناحیه را به‌دست می‌آورد.

- مختصات مرکز ستاره را با میانگین گرفتن از شروع و پایان محدوده‌ها محاسبه می‌کند.

- مختصات مرکز ستاره را به لیست star_centers اضافه می‌کند.

8. for i, (x, y) in enumerate(star_centers):

print(f"Star {i+1}: ({x}, {y})")

- مختصات مراکز ستاره‌ها را با شماره‌گذاری و در فرمت (x, y) چاپ می‌کند.

خروجی :

```
Number of stars detected: 11
Coordinates of stars:
Star 1: (90, 131)
Star 2: (147, 135)
Star 3: (155, 143)
Star 4: (157, 153)
Star 5: (57, 171)
Star 6: (118, 206)
Star 7: (22, 216)
Star 8: (154, 218)
Star 9: (156, 260)
Star 10: (233, 271)
Star 11: (225, 297)
```