به نام خدا

تمرین سوم – بینایی کامپیوتر **نیایش خانی ۹۹۵۲۱۲۳۵**

سوال ۱)

a) با استفاده از روابط زیر می توان بردار گرادیان را محاسبه کرد :

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

b) این بردار اطلاعاتی دربارهٔ شیب و تغییرات شدت رنگ در تصویر را فراهم می‌کند، که در تشخیص لبه‌ها، تشخیص کنتورها، استخراج ویژگی‌ها، ردیابی اشیاء، تشخیص الگو، اصلاح تصاویر، و بسیاری از وظایف پردازش تصویر دیگر مورد استفاده قرار می‌گیرد.
به عنوان مثال برای تشخیص لبه‌ها (*Edge Detection*) با محاسبه گرادیان تصویر، می‌توان لبه‌های تصویر را شناسایی کرده و از بیشترین تغییرات شدت رنگ برای تشخیص لبه‌ها استفاده کرد. در کاربرد بعدی برای اصلاح تصاویر (*Image Enhancement*) ، از گرادیان برای افزایش کیفیت تصویر، افزایش کنتراست، و کاهش نویز استفاده می‌شود.

c) اندازه گرادیان از فرمول زیر بدست می آید :

$$M(x,y) = \|\nabla f\| = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2} \approx |g_x| + |g_y|$$

d) فرمول محاسبه جهت گرادیان :

$$\alpha(x,y) = \text{dir}(\nabla f) = \text{atan2}(g_y, g_x)$$

e) مراحل استفاده از گرادیان در آشکارساز لبه Canny عبارت است از:

۱. هموار کردن تصویر با استفاده از فیلتر گاوسی برای کاهش نویز تصویر
۲. محاسبه گرادیان برای یافتن شیب های تصویر و جهت آنها
۳. حذف مقادیر غیر بیشینه : در این مرحله، بررسی می‌شود که آیا هر نقطه از تصویر در جهت گرادیان اصلی قرار دارد یا خیر. اگر نقطه‌ای از تصویر در جهت گرادیان اصلی قرار نگیرد (به صورت افقی یا عمودی یا دو نقطه مجاور خود قرار گرفته باشد) حذف می‌شود. این کار باعث می‌شود تنها نقاطی که به عنوان لبه واقعی شناخته شده‌اند باقی بمانند. در واقع هر پیکسل که در راستای گرادیان خود دارای مقدار غیر بیشینه باشد، حذف می شود.
۴. آستانه گذاری دو سطحی: در این مرحله، لبه‌های تکانه‌ای به صورت ناهموار و بدون ارتباط با لبه‌های اصلی حذف می‌شوند. همچنین لبه‌های نهایی با ترکیب و اتصال لبه‌های متصل در تصویر کشف می‌شوند. در واقع هر پیکسلی که اندازه گرادیان آن کوچکتر از T1 باشد غیر لبه و هر پیکسلی که اندازه گرادیان آن بزرگتر از T2 باشد لبه معرفی می شود. پیکسل هایی که اندازه گرادیان آنها بین T1 و T2 باشد در صورتی که به یک پیکسل لبه به صورت مستقیم متصل باشد، به عنوان لبه معرفی می شوند.

از مزایای Canny نسبت به روش‌های دیگر می توان به موارد زیر اشاره کرد :

- خطی بودن پاسخ *Canny*: توانایی بالایی در تشخیص لبه‌ها و کاهش نویز را داراست و نتایج آن به شکلی خطی و قابل پیش‌بینی به دست می‌آید.
- مقاومت بیشتر در برابر نویز نسبت به روش های دیگر

- دقت بالا: این الگوریتم به خوبی قادر به تشخیص لبه‌های حقیقی از لبه‌های مزیف است و دارای دقت بالایی است.

- تنظیم‌پذیری *Canny*: دارای پارامترهای قابل تنظیمی مانند اندازه فیلتر گوسی، حد نویز، و آستانه‌های هیسترزیس است که امکان کنترل دقیقتری بر روی فرآیند تشخیص لبه را فراهم می‌کند.

f) عمل‌گر لاپلاسین معمولاً به تنهایی برای تشخیص لبه استفاده نمی‌شود و به صورت ترکیبی با روش‌های دیگر مانند فیلترهای گرادیان مانند *Sobel* یا *Canny* استفاده می‌شود. این مسئله به دلایل زیر است:

۱. حساسیت به نویز : عمل‌گر لاپلاسین بسیار حساس به نویز است. هر گونه نویز در تصویر می‌تواند به عنوان لبه شناخته شده و نتیجه‌ای نامناسب را ایجاد کند. این حساسیت می‌تواند باعث کاهش دقت در تشخیص لبه‌های واقعی شود.
۲. کاهش اطلاعات مکانی : این عملگر تنها اطلاعات درجه دوم را از تصویر استخراج می‌کند و اطلاعات مکانی در مورد جهت لبه‌ها را ارائه نمی‌دهد. این اطلاعات مکانی برای تمایز بین لبه‌های واقعی و غیر لبه ها بسیار اهمیت دارد.
۳. از بین رفتن جزئیات : لاپلاسین تمایل به حذف جزئیات ریز در تصویر دارد. این امر می‌تواند به عدم تشخیص لبه‌های ناکافی و از دست رفتن جزئیات مهم در تصویر منجر شود.

سوال ۲)

```python
def draw_phase_amplitude(image):
    # Compute Fourier transform
    fourier_image = np.fft.fft2(image)

    # Shift the zero-frequency component to the center of the spectrum
    fourier_image_shifted = np.fft.fftshift(fourier_image)

    # Calculate the phase and amplitude of the image
    phase = np.angle(fourier_image_shifted)
    amplitude = np.abs(fourier_image_shifted)

    return phase, amplitude
```

a) در این کد تکمیل شده ما از تابع np.fft.fft2() برای محاسبه تبدیل فوریه ۲ بعدی تصویر ورودی استفاده می کنیم.

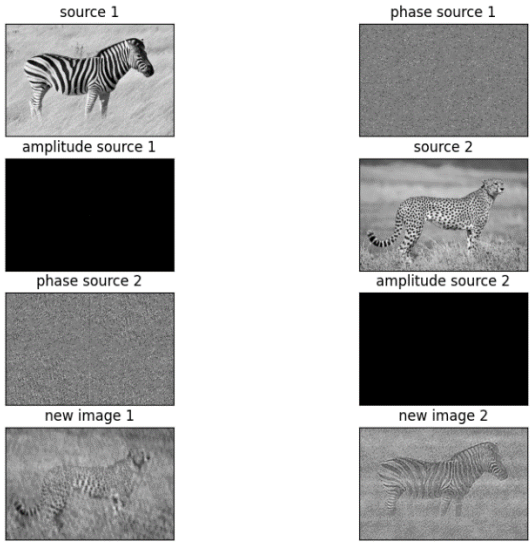
سپس، اطلاعات فاز و دامنه را از تبدیل فوریه با استفاده از توابع np.angle() و np.abs() استخراج می کنیم. در نهایت تصاویر فاز و دامنه را به صورت تاپل برمی گردانیم.

b) در بخش بعدی داریم :

```
def change_phase_domain(image1, image2):

    # Compute Fourier transforms
    f_transform1 = np.fft.fft2(image1)
    f_transform2 = np.fft.fft2(image2)

    # Compute magnitudes and phases
    mag1 = np.abs(f_transform1)
    mag2 = np.abs(f_transform2)
    phase1 = np.angle(f_transform1)
    phase2 = np.angle(f_transform2)

    # Substitute phase of image1 with phase of image2
    result1 = mag1 * np.exp(1j * phase2)
    result2 = mag2 * np.exp(1j * phase1)

    # Compute inverse Fourier transforms to get the new images
    img1 = np.fft.ifft2(result1).real
    img2 = np.fft.ifft2(result2).real

    return img1, img2
```

در این کد تکمیل شده مانند قبل تبدیل فوریه هر دو *image1* و *image2* و سپس، دامنه و فازهای هر دو تبدیل فوریه را محاسبه می کنیم. سپس فاز *image1* را با فاز *image2* جایگزین می کنیم و بالعکس. در نهایت، تبدیل فوریه معکوس تبدیل فوریه اصلاح شده را محاسبه می کنیم تا تصاویر جدید را به دست آوریم که به صورت تاپلی برگردانده می شوند.

خروجی به این شکل خواهد بود :



با تغییر فازها و دامنه های تصاویر و طبق مشاهدات خود از خروجی این بخش می توان دریافت که

همانطور که می دانیم فاز یک تصویر حاوی اطلاعاتی در مورد آرایش فضایی و جهت گیری ویژگی های درون تصویر است. نواحی با مقادیر فاز مشابه تمایل به بافت یا الگوهای مشابه دارند. تغییر فاز می تواند ظاهر بافت ها و الگوهای درون تصویر را تغییر دهد. بنابراین با عوض کردن فازهای دو تصویر، آرایش فضایی ویژگی‌ها در یک تصویر شبیه به تصویر دیگر خواهد شد که منجر به ترکیب ادراکی ویژگی‌های هر دو تصویر می‌شود.

از طرفی، دامنه یک تصویر نشان دهنده بزرگی یا شدت فرکانس های فضایی در تصویر است و تغییر دامنه با ثابت نگه داشتن فاز منجر به تغییر در روشنایی یا کنتراست کلی تصویر می شود. افزایش دامنه کنتراست را افزایش می دهد و تصویر را واضح تر نشان می دهد، در حالی که کاهش دامنه کنتراست را کاهش می دهد و تصویر را صاف تر نشان می دهد.

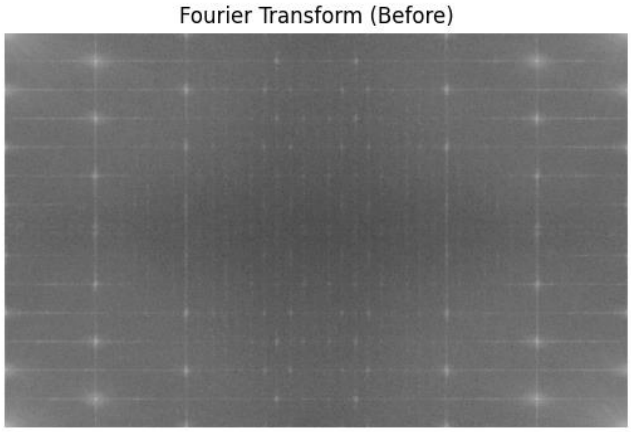
تغییر فاز و دامنه به طور همزمان منجر به تبدیل ترکیبی می شود که ساختار فضایی و ویژگی های شدت (intensity) تصاویر را تغییر می دهد.

به طور کلی، این امر منجر به ترکیب ادراکی ویژگی‌ها، texture ها و شدت‌های تصاویر اصلی می‌شود که منجر به یک تصویر ترکیبی جدید با ویژگی‌های تحت تأثیر هر دو نسخه اصلی می‌شود.

سوال ۳)

a) ابتدا تصویر را می خوانیم و تبدیل فوریه آن را محاسبه می کنیم.

```
# Read the image
image = cv2.imread('saffrun.jpg', cv2.IMREAD_GRAYSCALE)

# Compute the 2D Fourier transform
fft_image = np.fft.fft2(image)

# Visualize the magnitude spectrum
magnitude_spectrum = np.log(1 + np.abs(fft_image))
plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('Magnitude Spectrum (Before)')
plt.axis('off')
plt.show()
```

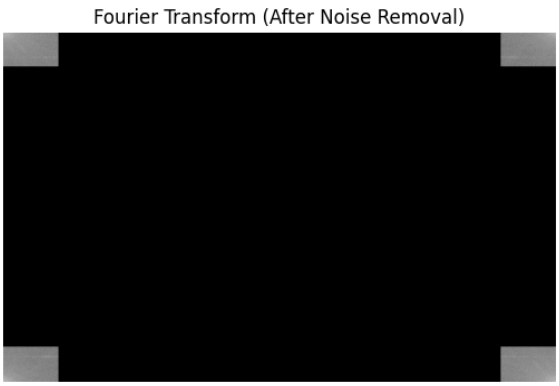


در اینجا خروجی مشاهده شده تبدیل فوریه تصویر قبل از حذف نویز است.

سپس مطابق کد زیر k را به عنوان درصد نقاط نویزی که باید حذف شوند در نظر می گیریم. سپس بخشی از ضرایب فوریه را صفر می کنیم تا نقاط نویز حذف شوند.

```
# Define the cutoff fraction for the noisy points to be removed
k = 0.1  # Adjust this value based on the amount of noise to be removed

# Identify the dimensions of the image
rows, cols = image.shape

# Apply the filter to remove noisy points
fft_image[int(rows * k):int(rows * (1 - k))] = 0
fft_image[:, int(cols * k):int(cols * (1 - k))] = 0
# Visualize the fourier transform after noise removal
fourier_transform_filtered = np.log(1 + np.abs(fft_image))
```

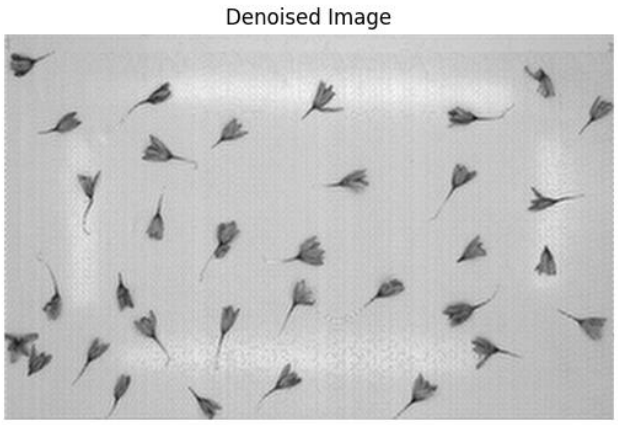
فوریه تصویربعد از حذف نویز به این شکل خواهد بود :



حال تبدیل فوریه معکوس را محاسبه می کنیم تا تصویر حذف شده را بدست آوریم.

```
# Compute the inverse Fourier transform to obtain the denoised image
denoised_image = np.fft.ifft2(fft_image).real
denoised_image = np.uint8(denoised_image)
```

در نهایت تصویر اصلی بعد از حذف نویز را نمایش می‌دهیم.

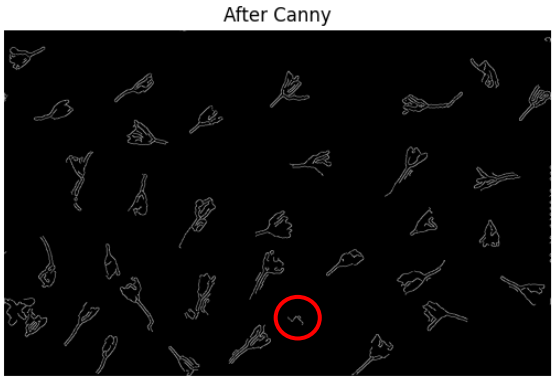


b) با استفاده از کد زیر لبه ها را شناسایی می کنیم.

```
edges_detected_image = cv2.Canny(np.uint8(denoised_image), 50, 100);
plt.imshow(edges_detected_image, cmap='gray')
plt.title('After Canny')
plt.axis('off')
plt.show()
```

همانطور که می دانیم آستانه پایین حداقل مقدار گرادیان را تعیین می کند و باید آستانه پایینی را انتخاب کنیم که به اندازه کافی پایین باشد تا لبه های ضعیف را تشخیص دهد اما به اندازه کافی بالا باشد تا نویز و گرادیان های ضعیف را سرکوب کند. همچنین آستانه بالا حداکثر مقدار گرادیان را تعیین می کند که به عنوان یک لبه قوی در نظر گرفته خواهد شد . باید طوری انتخاب شود که به اندازه کافی بالا باشد تا لبه های قوی و ویژگی های قابل توجه تصویر را ثبت کند.

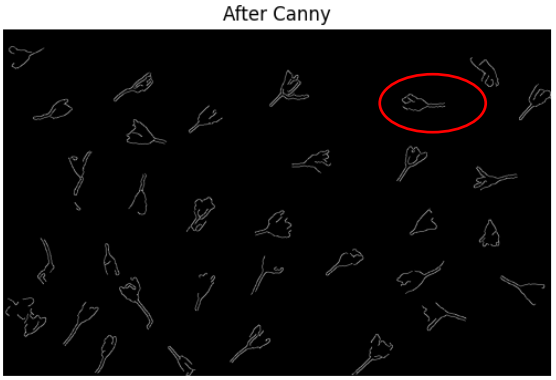
با انجام چند آزمون و خطا می توانیم به عدد های مناسب برسیم. برای مثال برای آستانه پایین اگر مقدار ۲۰ یا ۳۰ را در نظر بگیریم، در محل مشخص شده نویز را لبه در نظر گرفته و دچار خطا می شود.



از طرفی اگر برای آستانه بالا مقدار ۱۵۰ را در نظر بگیریم، یکسری از لبه ها تشخیص داده نمی شوند.

برای مثال در نقطه مشخص شده، لبه به طور کامل تشخیص داده نشده است.

در نهایت بعد از امتحان اعداد مختلف و مقایسه با تصویر اصلی به اعداد ۵۰ و ۱۰۰ می‌رسیم.



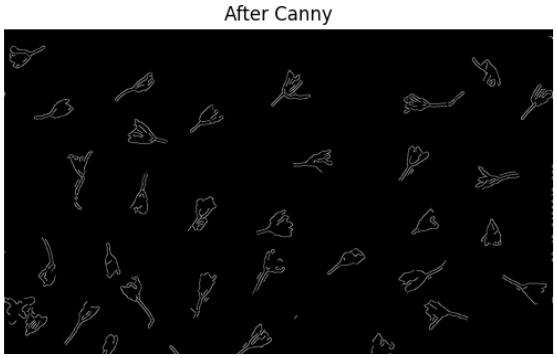


Figure ۱ - نتیجه نهایی

c) با استفاده از کد زیر گرادیان را محاسبه می کنیم.

```
# Compute the gradient in the x and y directions
gradient_x = cv2.Sobel(edges_detected_image, cv2.CV_64F, 1, 0, ksize=5)
gradient_y = cv2.Sobel(edges_detected_image, cv2.CV_64F, 0, 1, ksize=5)

# Compute the magnitude and direction of the gradient
magnitude, direction = cv2.cartToPolar(gradient_x, gradient_y)

# Compute the direction of the gradients using the 2*arctan function
gradient_direction = np.arctan2(gradient_y, gradient_x)

# If you want to convert the result from radians to degrees use below code:
# gradient_direction = np.degrees(2 * np.arctan(gradient_y / (gradient_x +
0.0001)))  # Adding a small value to avoid division by zero

print(gradient_direction)
```

تابع *cv2.Sobel* برای محاسبه مشتق تصویر در جهت های *x* و *y* که به ترتیب شیب در جهت *x* و *y* هستند استفاده می شود. *cv2.CV_64F* برای تعیین عمق تصویر مقصد و آرگومان های ۱، ۰ مشخص می کنند که تابع باید مشتق را در جهت *x* محاسبه کند، و آرگومان های ۰، ۱ مشخص می کنند که تابع باید مشتق را در جهت *y* محاسبه کند. آرگومان *ksize=5* اندازه هسته *Sobel* توسعه یافته را مشخص می کند که باید ۱، ۳، ۵ یا ۷ باشد.
سپس با استفاده از تابع *np.arctan2* جهت گرادیان ها را محاسبه می کنیم. این تابع زوایای بین $-\pi$ و π را بر حسب رادیان برمی گرداند . (برای تبدیل نتیجه از رادیان به درجه می توان از تابع *np.degrees* استفاده کرد.

d) با توجه به خروجی به دست آمده در قسمت قبل، در جهت تندترین فرود (یعنی جهت گرادیان منفی) حرکت می‌کنیم تا به ساقه برسیم. هنگامی که جهت گرادیان تغییر قابل توجهی را نشان می دهد (رسیدن به منطقه ساقه) توقف می‌کنیم که این نقطه توقف احتمالاً نقطه برش ساقه از گلبرگ است.

سوال ۴)

a) سه نمونه از کاربردهای تبدیل فوریه :
فشرده سازی تصویر:

- تحلیل فوریه، به ویژه تبدیل فوریه گسسته (*DFT*)، به طور گسترده ای در تکنیک های فشرده سازی تصویر مانند *JPEG* و *JPEG2000* استفاده می شود.
- *DFT* به تبدیل یک تصویر از حوزه فضایی به حوزه فرکانس کمک می کند، جایی که انرژی در ضرایب کمتری (مولفه های فرکانس پایین) متمرکز می شود.
- با دور انداختن یا کمی کردن اجزای فرکانس بالا (که نمایانگر جزئیات دقیق و نویز هستند)، می توان تصویر را با حفظ کیفیت بصری به طور موثر فشرده کرد.
- این فشرده سازی فضای ذخیره سازی و پهنای باند مورد نیاز برای انتقال تصاویر از طریق شبکه ها را کاهش می دهد و به اشتراک گذاری و ذخیره سازی تصویر را کارآمدتر می کند.

بازیابی تصویر:

- فوریه در تکنیک های بازیابی تصویر برای حذف نویز و مصنوعات و افزایش کیفیت تصویر استفاده می شود.
- در تکنیک هایی مانند فیلتر وینر و فیلتر معکوس، تصویر تخریب شده با استفاده از *DFT* به حوزه فرکانس تبدیل می شود.
- نویز و سایر اجزای ناخواسته در حوزه فرکانس فیلتر می شوند و با انجام یک *DFT* معکوس تصویر بازیابی شده به دست می آید.
- آنالیز فوریه جداسازی اجزای سیگنال و نویز را امکان پذیر می کند و کاهش موثر نویز و بازیابی جزئیات تصویر را تسهیل می کند.

فیلتر کردن تصویر:

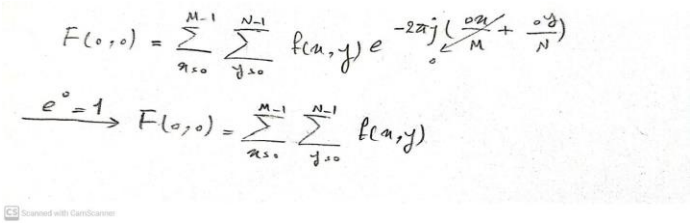
- تحلیل فوریه اغلب در فیلترهای حوزه فرکانس مانند فیلترهای پایین گذر، بالا گذر و باند استفاده می شود. این فیلترها برای حذف نویز، بهبود ویژگی‌های خاص یا انجام تغییرات دیگر روی یک تصویر استفاده می‌شوند.
- تبدیل فوریه برای تبدیل تصویر از حوزه فضایی به حوزه فرکانس، جایی که عملیات فیلتر کردن انجام می شود، استفاده می شود .
- مزیت این رویکرد این است که عملیات خاصی مانند کانولوشن که از نظر محاسباتی در حوزه فضایی گران هستند، به عملیات ضرب ساده در حوزه فرکانس تبدیل می شوند.

b) برای محاسبه تبدیل فوریه از فرمول زیر استفاده می‌کنیم:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)}$$

بنابراین داریم :

در نتیجه حاصل برابر است با مجموع مقادیر f(x,y) .



سوال ۶) در فرمول مورد نظر

- k نشان دهنده تعداد تکرارهاست.
- p احتمال مورد نظر برای یافتن پارامترهای صحیح است (در اینجا، ۰/۹۹).
- w نسبت لبه‌های مربوط به دایره به کل لبه‌ها است (در اینجا، ۰/۴).
- s تعداد نقاط مورد نیاز برای تعیین پارامترهاست (برای یک دایره، معمولاً ۳ نقطه لازم است).

$$k = \frac{log(1-p)}{log(1-w^3)} = \frac{log(1-0.99)}{log(1-0.4^3)} = \frac{-2}{-0.0287} \approx 70$$

سوال ۷)

a) سه جنبه برای مقایسه بین تبدیل Hough و الگوریتم LSD (Line Segment Detector) برای تشخیص خط:

۱. Methodology :

- الگوریتم Hough : بر اساس یک طرح رأی گیری است که در آن هر نقطه لبه به خطوط بالقوه ای که از آن عبور می کنند رأی می دهد. از فضای انباشته برای نشان دادن این آرا استفاده می شود و قله ها در فضای انباشته با خطوط بالقوه در تصویر مطابقت دارند.
- الگوریتم LSD: مستقیماً روی پیکسل های لبه عمل می کند و بخش های خط را از تصویر استخراج می کند. با بررسی ویژگی های محلی پیکسل های لبه بدون محاسبه صریح فضای پارامتر، بخش های خط مستقیم را تشخیص می دهد.

۲. دقت و کارایی :

- تبدیل Hough : در حالی که تبدیل Hough در برابر نویز قوی است و می تواند خطوط را با دقت بالایی تشخیص دهد، به دلیل نیاز به جستجو در فضای پارامتر، می تواند از نظر محاسباتی پر هزینه باشد، به خصوص برای تصاویر بزرگ یا پیکربندی های خط پیچیده.
- الگوریتم LSD : ال اس دی به دلیل کارایی و سرعت آن معروف است. حتی در صورت وجود نویز و درهم و برهمی نیز می تواند قطعات خطوط را به دقت تشخیص دهد. این به طور مستقیم بر روی پیکسل های لبه بدون جستجوی صریح فضای پارامتر عمل می کند، که آن را از نظر محاسباتی کارآمد می کند.

۳. تشخیص خطوط چندگانه :

- تبدیل Hough : تبدیل Hough قادر است چندین خط را به طور همزمان در یک تصویر تشخیص دهد. با تجزیه و تحلیل فضای انباشته، می توان قله های مربوط به خطوط مختلف را شناسایی کرد که امکان تشخیص خطوط متعدد با جهت ها و موقعیت های مختلف را فراهم می کند.

- الگوریتم LSD: در حالی که الگوریتم LSD در تشخیص تک تک بخش های خط کارآمد است، ممکن است برای شناسایی چندین خط در یک تصویر به مراحل پردازش یا پس پردازش اضافی نیاز داشته باشد. از آنجایی که LSD به‌جای کل خطوط بر استخراج بخش‌های خط تمرکز دارد، ممکن است نیاز باشد که به صورت تکراری یا ترکیبی با الگوریتم‌های خوشه‌بندی برای شناسایی دقیق چندین خط استفاده شود.