Q1)

(a) The gradient vector field $\vec{\nabla} I(x,y)$ of an image $I(x,y)$ is a tuple of partial derivatives associated with each point in the image: $\vec{\nabla} I(x,y) \equiv \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$

(b) This vector field can be used to detect local edges in the image, estimating both their strength and their direction.

(c) The gradient magnitude, estimating edge strength, is: $\|\vec{\nabla} I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$

(d) The gradient direction (orientation of an edge) is estimated as: $\theta = \tan^{-1}\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$

(e) In the Canny edge detector the following steps are applied, resulting in much cleaner detection of the actual boundaries of objects, with spurious edge clutter eliminated:

1. First the image is smoothed with a Gaussian filter to reduce noise.
2. Then the gradient vector field $\vec{\nabla} I(x,y)$ is computed across the image. The partial derivatives in $\vec{\nabla} I(x,y)$ can be estimated as the first finite differences in x and in y.
3. An "edge thinning" technique, non-maximal suppression, eliminates spurious edges. An edge is represented by a single pixel at which the gradient is maximal.
4. Applying a double threshold to the gradient magnitude enables a triage of edge data, labelling it as strong, weak, or suppressed.
5. A connectivity constraint is applied, by "tracking" detected edges across the image. Edges that are weak and not connected to strong edges are eliminated.

(f) The Laplacian operator is not typically preferred for edge detection because it is highly sensitive to noise, uses a single threshold that may result in inconsistent edge detection, and does not provide edge orientation information, unlike the Sobel and Canny operators which address these issues more effectively.

Q2).اطلاعات و محتویات تصویر در فاز آن ذخیره میشود

Please review the notebook file Q2.ipynb.

Q3) Please review the notebook file Q3.ipynb.

Q4)

a)

Any three from the following list would do:

1. Convolution of an image with some operator, for example an edge detection operator or feature detecting operator, is ubiquitous in computer vision. Convolution is computationally costly and slow if done "literally," but it is very efficient if done instead in the Fourier domain. One merely needs to multiply the Fourier transform of the image by the Fourier transform of the operator in question, and then take the inverse Fourier transform to get the desired result. For kernels larger than about (5×5), the benefit is that the Fourier approach is vastly more efficient.
2. The Fourier perspective on edge detection shows that it is really just a kind of frequency-selective filtering, usually high-pass or bandpass filtering. For example, applying the ∇^2 second-derivative operator to an image is equivalent to multiplying its Fourier transform by a paraboloid, $\mu^2 + \nu^2$, which discards low frequencies but emphasises high frequencies, in proportion to their square.
3. Texture detection, and texture segmentation, can be accomplished by 2D spectral (Fourier) analysis. Textures are well-defined by their spatial frequency and orientation characteristics, and these indeed are the polar coordinates of the Fourier plane.
4. Motion can be detected, and its parameters estimated, by exploiting the "Spectral co-planarity theorem" of the 3-D spatio-temporal Fourier transform.
5. Active contours as flexible boundary descriptors ("snakes") can be implemented through truncated Fourier series expansions of the boundary data.

b)

In the frequency domain, it is equivalent to summing the values of all pixels in the image. In other words, the value at point (0, 0) equals the sum of all pixel values in the image.

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

$$F(0,0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi(0)}$$

$$F(0,0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)$$

Q5) Please review the notebook file Q5.ipynb.

Q6)

چون برای ترسیم دایره به سه نقطه نیاز داریم، اگر W نسبت به تعداد نقاط inlier به تمام نقاط باشد، احتمال تشکیل شدن یک مجموعه‌ی سه تایی کاملا از نقاط inlier تشکیل شود w^3 خواهد بود .احتمال یافتن مجموعه ای از نقاط بدون outlier است و k تعداد تکرار الگوریتم. داریم:

$$1-p = (1-w^3)^k \quad => \quad k = \frac{\log(1-p)}{log(1-w^3)} = \frac{\log(1-0.99)}{log(1-0.4^3)} = \frac{\log(0.01)}{log(0.064)} = 69.6$$

پس بعد از هفتاد بار اجرا کردن الگوریتم، با احتمال بیشتر از 99 درصد پارامترهای مناسبی داریم

Q7)

الف)

1. پیچیدگی محاسباتی: الگوریتم Hough پیچیدگی محاسباتی بالاتری نسبت به LSD دارد، زیرا برای محاسبه پارامترهای خط در فضای پارامتری Hough، نیاز به محاسبات زیادی دارد. از طرف دیگر، LSD با استفاده از روش های بهینه سازی شده، پیچیدگی محاسباتی کمتری دارد.

2. کارایی در تصاویر پر نویز: الگوریتم Hough مقاوم‌تر در برابر نویز است و می‌تواند خطوط را حتی در تصاویر پر نویز نیز تشخیص دهد. در مقابل، LSD نسبت به نویز حساس‌تر است و ممکن است در تصاویر پر نویز، عملکرد ضعیف‌تری داشته باشد.

3. انعطاف‌پذیری: الگوریتم Hough قادر است انواع مختلفی از شکل‌ها مانند دایره‌ها، بیضی‌ها و منحنی‌ها را نیز تشخیص دهد، در حالی که LSD فقط برای تشخیص خطوط طراحی شده است. بنابراین، الگوریتم Hough انعطاف‌پذیری بیشتری دارد.

4. هزینه محاسباتی: به طور کلی، الگوریتم LSD هزینه محاسباتی کمتری نسبت به Hough دارد، زیرا از روش‌های بهینه‌سازی شده برای تشخیص خطوط استفاده می‌کند. این امر باعث می‌شود که LSD برای کاربردهای زمان واقعی مناسب‌تر باشد.

در نهایت، انتخاب بین الگوریتم Hough و LSD بستگی به نیازهای خاص کاربرد دارد. اگر نیاز به تشخیص انواع مختلف شکل‌ها و مقاومت در برابر نویز وجود داشته باشد، الگوریتم Hough گزینه مناسب‌تری است. اما اگر سرعت و کارایی محاسباتی مهم‌تر باشد، LSD انتخاب بهتری خواهد بود.

ب)
پاسخ در نوت‌بوک Q7

د) پاسخ در نوت‌بوک Q7

Q8) Please review the notebook file Q8.ipynb.