

Kajol Tanesh Shah

A20496724

CS512 Spring 2022

Assignment 2

Q1. a) SNR in an image:

$$\text{SNR} = \frac{E_s}{E_n} = \frac{\sigma_s^2}{\sigma_n^2} = \frac{1}{n} \sum_{i,j} (I(i,j) - \bar{I})^2$$

where, σ_n^2 is the variance of the noise.

σ_n^2 = variance for multiple frames
of a static scene or variance
in a uniform image region.

E_s = Energy of signal

E_n = Energy of noise

SNR is the ratio of the desired
signal to background noise.

Q1.b) (a) Gaussian noise

1. We see Gaussian noise is the fluctuation that we can see in the images as we move to the next levels.
2. It occurs due to sensor limitations while capturing image under low-light conditions.
3. Gaussian filter handles gaussian noise better.

(b) Impulsive noise

1. Impulsive noise can be caused due to the defect in the sensors and also called as salt and pepper noise.
2. It corrupts the images and replaces some of the pixels in the original image.
3. Median filter handles impulsive noise better.

Averaging filter is sensitive to outliers whereas median filter is not. So, median filter handles impulsive noise better.

Q1.c

Let the image be $I = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$

2	2	2	2	2
2	2	2	2	2
2	2	2	2	2
2	2	2	2	2
2	2	2	2	2

Convolution filter \Rightarrow

1	1	1
1	1	1
1	1	1

If we compute the filter with one window of the image, we will get

$$1*2 + 1*2 + 1*2 + 1*2 + 1*2 + 1*2 + 1*2 = 18$$

In convolution, we replace the computed value at the center of 3×3 window.

So, after applying convolution to the image I , we will get

18	18	18	18	18
18	18	18	18	18
18	18	18	18	18
18	18	18	18	18
18	18	18	18	18

Q1.d) Convolution is a linear filter and according to convolution properties

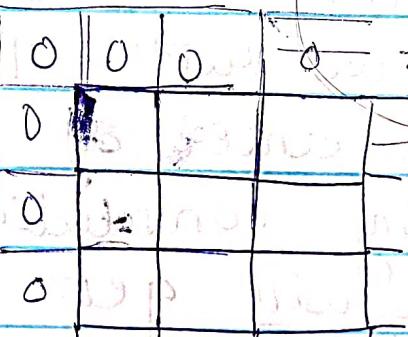
$$\frac{d}{dx}(f * g) = \frac{d}{dx} f * g = f * \frac{d}{dx} g$$

where f is the filter & g is the image
In this way, the operation can be applied more efficiently.

Q.1.c. The three different ways to handle boundaries convolution dice during

① zero padding

add zeros around the image when we use the filter and number of zeros depends on size of the filter

e.g.  zeros padded.

0	0	0
0	5	6
0	7	8

image

0	0	0
0	10	11
0	12	13

image

② Mirror / replicate - better than zero

replicated values	5	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000

③ Ignore the outside values of the filter or boundaries (if image is large and filter is small)

Q1.f Smoothing filter = $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Sum of all the entries = $\frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9}$

$$+ \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} = \frac{9}{9} = 1.$$

The sum should be equal to 1 so as to keep the mean value of the filtered image.

$$\text{If } I_G = I * G = \sum_{i=0}^M \sum_{j=0}^N I(i,j) e^{-\frac{i^2+j^2}{2\sigma^2}}$$

$$= \sum e^{-\frac{i^2}{2\sigma^2}} \sum I(i,j) e^{-\frac{j^2}{2\sigma^2}}$$

$$= (I * G_x) * G_y$$

1D gaussian 1D gaussian

cols

rows

One 2D pass will perform $M \times N \times m^2$ operations

2 1D passes will perform $2 \times M \times N \times m$ operations

So, implementing 2D Gaussian using two 1D convolution filters is more efficient.

It is possible to implement any 2D filters in this way only if the filter is linear and separable. Only then, the 2D convolution filter can be decomposed into 1D convolutions.

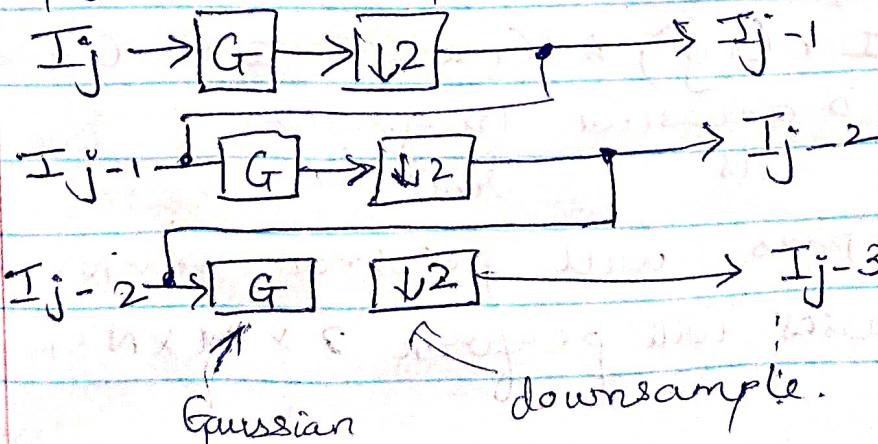
(Q.1.h)

$$\sigma = 2$$

$$m \geq 5\sigma + 1 + 1 + 1 \\ \therefore m \geq 5 \times 2 + 10 = 20$$

(Q.1.i) We use image pyramids for multiple scale analysis. The purpose of producing such image pyramids is to reduce the size of the image.

Below is the way in which Gaussian pyramid is produced in the following way



At each level, half of the size of the image is considered.

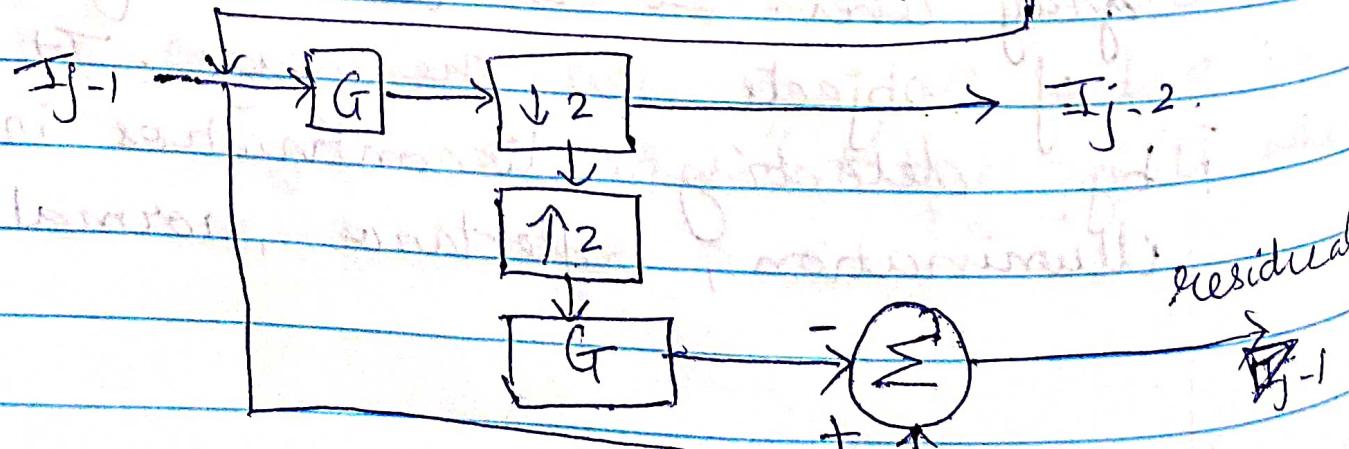
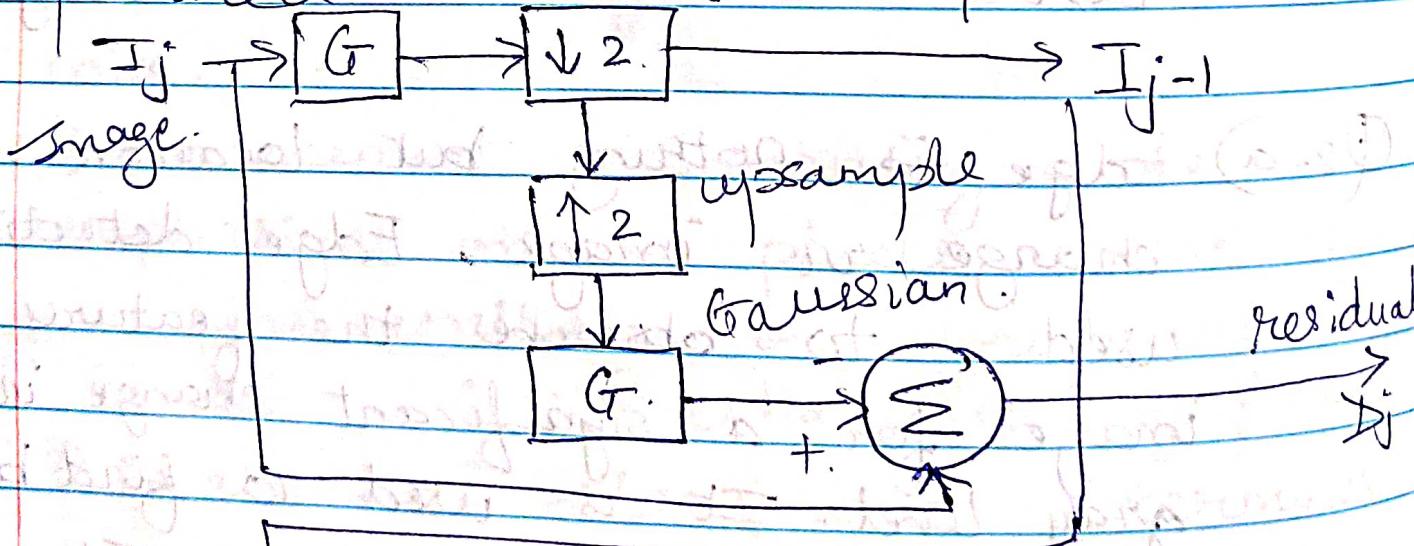
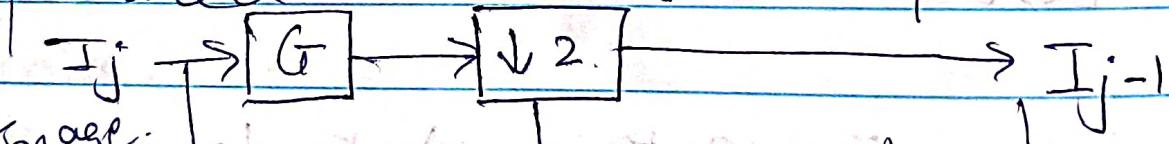
$$J = \log_2 m$$

Total number of pixels in a pyramid will be (Additional processing).

$$m^2 + \frac{1}{4}m^2 + \frac{1}{16}m^2 + \dots < \frac{4}{3}m^2$$

for single image

(Q.i.j) Below is how Laplacian pyramid is produced



The Laplacian pyramid is useful for compression of image (small residuals)

In Laplacian, first we convolve the

(2.a) image using Gaussian (smoothing), then downsample ~~at~~, upsample (back to original image size) and again convolve (Gaussian). After that, we compute the difference between the original image and the new ^{reduced} image (which is a little destroyed due to the operations)

D_j is the difference in the images.

We continue this till we reach the last level which is the image I_0 .

Q2.a) Edge is nothing but location with change in image. Edge detection is used to observe the features of an image for a significant change in the gray level. It is used to find boundaries of objects in the image. It works by detecting discontinuities in depth, illumination, reflectance, normal, etc.

The desired properties of edge detection

are:-

- 1) corresponding to scene elements
- 2) invariant
- 3) reliable detection

Q2.b) Basic steps of edge detection

- 1) Smoothing - Smoothing is done to reduce noise but without affecting the edges or blurring them. This improves the performance of an edge detector as they are sensitive to noise.
- 2) Enhancement - Enhancing the edges emphasizes on the pixels where there is edge contrast or significant change in the image. To enhance quality of edges
- 3) Localization - Localization determines the exact location of an edge in the image. We can detect the points which lie on the edge

(Q2.C) To compute change in image intensity, we need image gradient. Gradient vector points in the direction of maximum change. The magnitude of the gradient provides information about the strength of the image edge. We use gradients for detecting edges in image and to find contours & outlines of objects in images. The gradient of an image measures how it is changing; the magnitude tells how quickly the image is changing while the direction of the gradient tells us the direction in which the image is changing.

- ① Sobel filter (for computing image gradient). In this filter, we first do smoothing and then take derivatives.

$$\Delta x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\Delta y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

② Laplace filter or LOG

In this, we first do smoothing using Gaussian and then apply Laplacian

$$\Delta I = \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = I_{xx} + I_{yy}$$

$$I_{xx} = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} * I$$

$$I_{yy} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & 1 \end{bmatrix} * I$$

$$I_{xx} + I_{yy} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} * I$$

Q2. In Sobel filter, we first do smoothing and then take derivative

$$I_x = I * \left(G * \frac{\partial}{\partial x} \right) \quad I_y = I * \left(G * \frac{\partial}{\partial y} \right)$$

$$\therefore \Delta x = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 2 \end{bmatrix}$$

$$\Delta y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$(Q2.c) \quad I_x = I * G_x \quad \text{and} \quad I_y = I * G_y$$

By using separable property of Gaussian

$$I_x = I * G'_x * G_y \leftarrow 1D \text{ convolution}$$

~~1D convolution with horizontal Gaussian derivative with vertical Gaussian~~

$$I_y = I * G'_y * G_x \leftarrow 1D \text{ convolution}$$

~~1D convolution with vertical Gaussian derivative with horizontal Gaussian~~

$$I_y = \boxed{\text{Image}} * \boxed{\text{derivative}} * \boxed{\text{smooth}}$$

derivative

$$I_y = \boxed{\text{Image}} * \boxed{\text{smooth}} \quad I_x = \boxed{\text{Image}} * \boxed{\text{smooth}} * \boxed{\text{derivative}}$$

$$G(x) = e^{-\frac{x^2}{2\sigma^2}}$$

$$\therefore G'(x) = -\frac{2x}{2\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

$$G'(y) = -\frac{2y}{2\sigma^2} e^{-\frac{y^2}{2\sigma^2}}$$

\therefore If $\sigma = 2$; then I_x & I_y

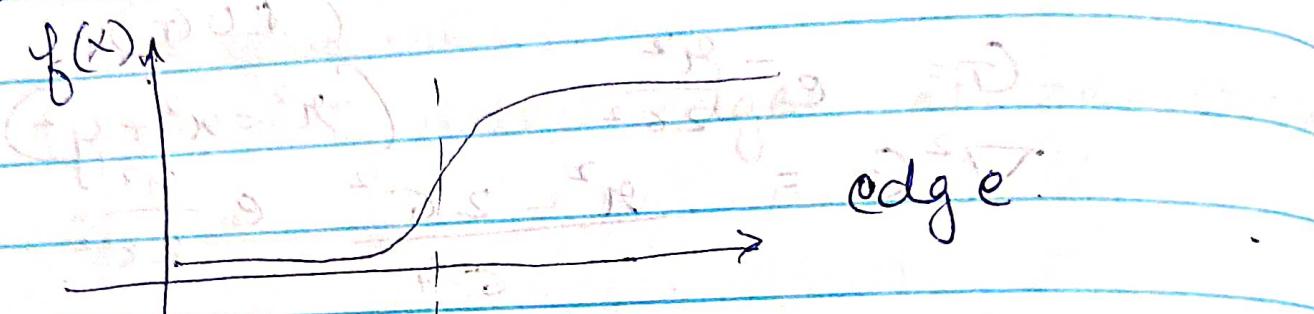
$$I_x = I * \left(-\frac{x}{2\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \right) = I * \left(-\frac{x}{4} e^{-\frac{x^2}{8}} \right)$$

$$I_y = I * \left(-\frac{y}{4} e^{-\frac{y^2}{8}} \right)$$

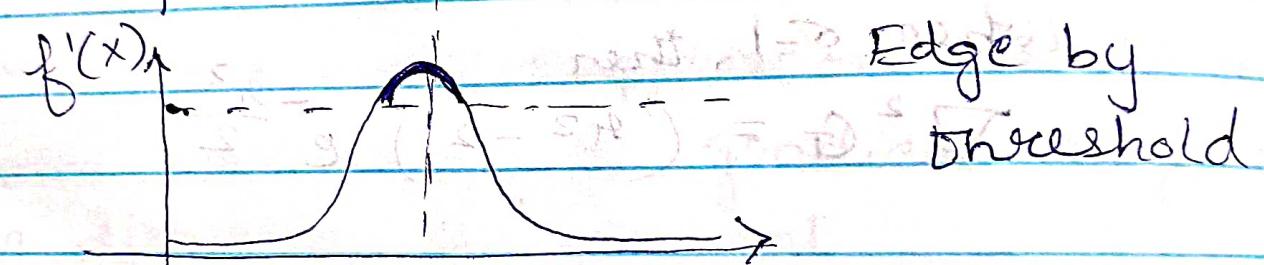
Q2. D)

Using Laplacian of Gaussian, we can detect edges using the first and second order derivative of the image.

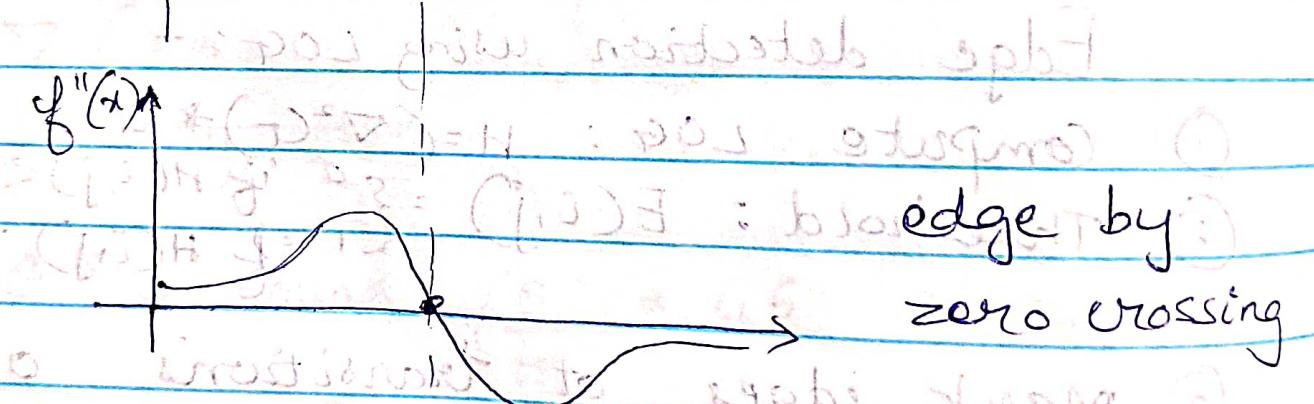
$$f(x)$$



$$f'(x)$$



$$f''(x)$$



Using first order derivative, we can localize the edge by threshold.

Using second order derivative, we can localize the edge by zero

crossing.

(Q2.g)

$$G = e^{-\frac{r^2}{2\sigma^2}}$$

LOG ?

$$H = \nabla^2(I * G) = \nabla^2(G * I)$$

(LOG)

$$G = e^{-\frac{r^2}{2\sigma^2}} \quad (r^2 = x^2 + y^2)$$

$$\nabla^2 G = \frac{r^2 - 2\sigma^2}{\sigma^4} e^{-\frac{r^2}{2\sigma^2}}$$

when $\sigma=1$, then

$$\nabla^2 G = (r^2 - 2) e^{-\frac{r^2}{2}}$$

Edge detection using LOG:-

①

Compute LOG : $H = (\nabla^2 G) * I$

②

Threshold : $E(i,j) = \begin{cases} 0 & \text{if } H(i,j) < 0 \\ 1 & \text{if } H(i,j) \geq 0 \end{cases}$

③

Mark edges at transitions $0 \rightarrow 1, 1 \rightarrow 0$
(Scan left to right and right to bottom)

(Q2.h)

Canny edge detection algorithm uses directional derivative to detect edges whereas standard edge detection algorithm does not. Canny edge detection detects edges at zero crossings of second order directional derivative taken along the gradient. Also, it will attempt

detection only if gradient magnitude is large enough ($|n| > \tau$). This is the condition for detecting edges.

$n = \text{gradient}$

if $|n| > \tau$ detect edges at zero crossing

$$\text{of } \frac{\partial^2}{\partial n^2} (I * G)$$

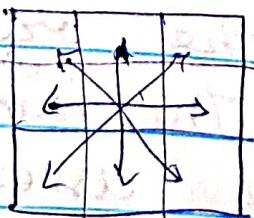
Q2.i) Non maximum suppression is to get local maximum of gradient magnitude in direction of gradient.

$$\nabla(I * G) = (I_x, I_y)$$

$$\theta = \tan^{-1} \left(\frac{I_y}{I_x} \right)$$

$$\theta^* = \text{Round} \left(\frac{\theta}{45} \right) * 45$$

$$E(i, j) = \begin{cases} 1 & \text{if } \nabla(I * G) \text{ is a local maximum} \\ 0 & \text{otherwise} \end{cases}$$



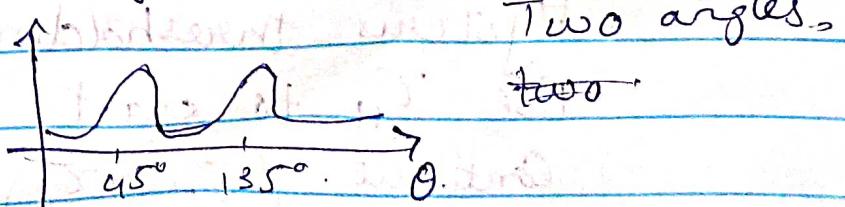
compare neighbours at their discretization.

Hysteresis thresholding
Use τ_H to start tracking and τ_L to continue ($\tau_H > \tau_L$)

- ① Initialize array of visited places
pixels $v(i, j) = 0$
- ② Scan image T-B, L-R:
if $|v(i, j)| \neq 0 \text{ and } |\nabla I| > \tau_h$ start tracking
an edge
- ③ Search for additional neighbours in
direction orthogonal to ∇I such that
 $|\nabla I| > \tau_e$

(Q3&d). Corner detection is used to find points of change in an image. It is used to track or characterize objects and characterize local windows. A corner is detected if there is more than one direction (in orientation histogram) of an image. So, we find two distinct directions in the orientation histogram for corner. The number of principal direction is assessed using orientation histograms.

e.g. for corner



We get a bunch of vectors and we try to find a principal direction which aligns best with all vectors. We use PCA to assess number of principal directions.

- ① We first find correlation matrix of gradients in local windows
- ② Find eigen values of correlation matrix
- ③ Detect corner in window if eigen values are sufficiently large

Q3.b) In PCA, to find principal directions for gradient orientations, we find directions that will minimize the projection on them so that they are orthogonal to previous directions.

Using PCA, we find direction v such that projection of $\{g_i\}$ onto v is minimized.

$$E(v) = \sum_i (g_i \cdot v)^2 = \sum_i (g_i^T v)(g_i^T v)$$

$$E(v) = v^T C v$$

where v is a 2D vector & C is the correlation matrix.

$$C = \sum_i g_i g_i^T = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i \\ \sum x_i y_i & \sum y_i^2 \end{bmatrix}$$

$$\text{where } g_i = [x_i, y_i]^T$$

$$\therefore v^* = \underset{v}{\operatorname{argmin}} E(v).$$

$$\frac{d}{dv} E(v) = 0$$

$$\frac{d}{dx} E = 0$$

$$\therefore \nabla E(v) = 0$$

$$\frac{d}{dy} E = 0$$

$$2Cv = 0$$

Q3.c) correlation matrix $C = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i \\ \sum x_i y_i & \sum y_i^2 \end{bmatrix}$

$$C = \begin{bmatrix} 1+1+1+1+1 & 1+2+3 \\ 1+2+3 & 1+4+9+16+1+4+9 \end{bmatrix}$$

$$C = \begin{bmatrix} 5 & 6 \\ 6 & 44.82 \end{bmatrix} = \begin{bmatrix} 4 & 6 \\ 6 & 44 \end{bmatrix}$$

Q3.d). Corner is detected if

i.e. dot product of eigen values
should be larger than threshold.

$\lambda_1 \cdot \lambda_2 > T$

Q3.e). Non maximum suppression

- ① compute λ_1, λ_2 for all windows
- ② select windows with $\lambda_1, \lambda_2 > T$
and sort in decreasing order
- ③ select the top of the list as corner
and delete all other corners in its
neighbourhood from the list
- ④ stop once detecting $x\%$ of the
points as corners.

(Q3. f) Harris corner detection (3, 8)

$$G(C) = \frac{\det(C)}{\lambda_1 \cdot \lambda_2} - k \text{tr}^2(C)$$
$$= \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} - k(\lambda_1 + \lambda_2)^2$$

$$= \lambda_1 \lambda_2 - k(\lambda_1^2 + 2\lambda_1 \lambda_2 + \lambda_2^2)$$
$$= (\lambda_1 - 2k)\lambda_2 - k(\lambda_1^2 + \lambda_2^2)$$

$\lambda_1 \lambda_2(1 - 2k) \geq 0$ if $k = 0.5 \Rightarrow$ corner detection

$k(\lambda_1^2 + \lambda_2^2) = 0$ if $k = 0 \Rightarrow$ edge detection.

So, without computing λ_1 & λ_2 ,

if $k = 0.5$ $G(C)$ detects edges

if $k = 0$ $G(C)$ detects corners.

Harris corner detection will detect corners where $G(C)$ is high.

(Q3. g) Corner localization means given that there is a corner in a window, find its location. To determine if p is the corner connect each point x_i to p and project the gradient at x_i onto $(x_i - p)$.

$$E(P) = \sum_i (\nabla I(x_i) \cdot (x_i - P))^2$$

$$E(P) = \sum_i (x_i - P)^T (\nabla I(x_i) \nabla I(x_i)^T) (x_i - P)$$

$$\text{min}_P P^* = \underset{P}{\operatorname{arg\min}} E(P)$$

$$\nabla E(P) = 0 \text{ pd hessian splitting}$$

$$-2 \sum_i (\nabla I(x_i) \nabla I(x_i)^T) (x_i - P) = 0$$

$$\sum_i \nabla I(x_i) \nabla I(x_i)^T P = \sum_i \nabla I(x_i) \nabla I(x_i)^T x_i$$

$$P^* = \left(\sum_i \nabla I(x_i) \nabla I(x_i)^T \right)^{-1} \sum_i \nabla I(x_i) \nabla I(x_i)^T x_i$$

$$P^* = C^{-1} \sum_i \nabla I(x_i) \nabla I(x_i)^T x_i$$

location of corner & orientation

$C \rightarrow$ correlation matrix

If we detected corners in the window $\lambda_1, \lambda_2 > \tau \Rightarrow C$ must be non-singular.

So, for the formula to be true, C should be non-singular.

(Q3.h) Characterization of feature points using HOG.

- ① split each patch into cells, possibly overlapping
- ② Create orientation histogram in each cell (using edge or gradient directions, possibly weighted by distance from center or gradient magnitude).
- ③ Concatenate orientation histograms

Requirements for a good characterization of feature points:-

- ① translation variance \rightarrow local window
- ② rotation variance \rightarrow histograms
- ③ scale variance \rightarrow pyramid
- ④ illumination variance \rightarrow gradients

(Q3.i) Scale-Invariant Feature Transform (SIFT)

- ① Use weighted sum of gradient directions to create orientation histogram's in cells, then concatenate
- ② Align histogram based on dominant direction (rotation invariant)