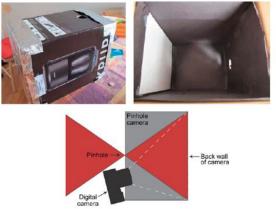
Cameras and Images

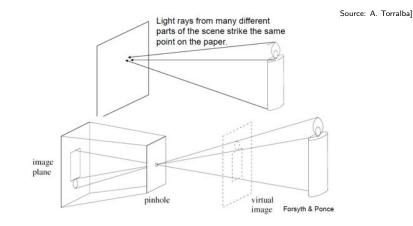
Pinhole Camera



[Source: A. Torralba]

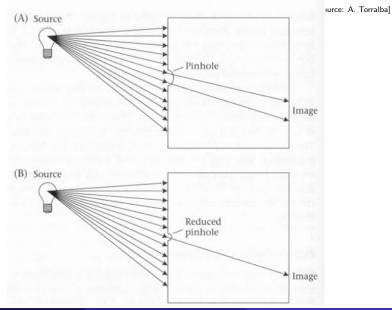
- Make your own camera
- http://www.foundphotography.com/PhotoThoughts/archives/2005/ 04/pinhole_camera_2.html

Pinhole Camera – How It Works



• The pinhole camera only allows rays from one point in the scene to strike each point of the paper.

Pinhole Camera – How It Works



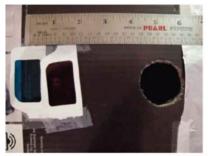
Pinhole Camera – Example

[Source: A. Torralba]





[Source: A. Torralba]





• You can make it stereo

Sanja Fidler

Pinhole Camera – Stereo Example

[Source: A. Torralba]

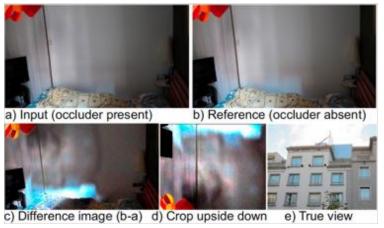


• Try it with 3D glasses!

Sanja Fidler

Pinhole Camera

[Source: A. Torralba]



- Remember this example?
- In this case the window acts as a pinhole camera into the room

Digital Camera



- A digital camera replaces film with a sensor array
- Each cell in the array is a light-sensitive diode that converts photons to electrons
- http://electronics.howstuffworks.com/cameras-photography/ digital/digital-camera.htm

Image Formation

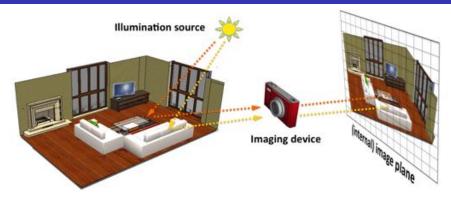


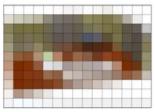
Image formation process producing a particular image depends on:

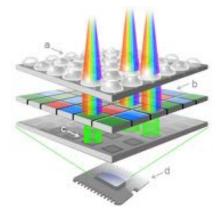
- lighting conditions
- scene geometry
- surface properties
- camera optics

Continuous image projected to sensor array



Sampling and quantization





http://pho.to/media/images/digital/digital-sensors.jpg

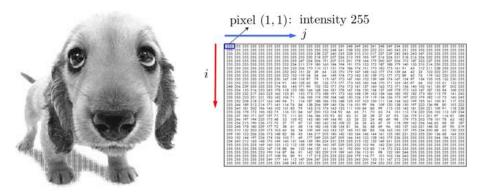
- Sample the 2D space on a regular grid
- Quantize each sample (round to nearest integer)

- Image is a matrix with integer values
- We will typically denote it with I



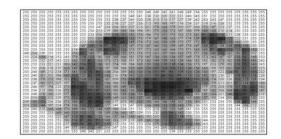
															248											255	255	255	25
		255					255											222				251				255	255	235	25
		265			255	255	255					237						217		239	242		243	247	255	255	255	255	25
299	239	255	399	255	595	595	255	255		232	210		227		212	178			200	224	231	218	216	236	202	599	255	335	
255	255	255	255	255	255	255	255	225	247	224	206	191	227	215	201	178	164	179	200	207	204	172	167	222	227	255	255	255	25
		255			225	255	255	224	211	218	199	160	294	194	191	170	152	172	107	100	178	140	155	272	214	229	222	255	25
255			255		399	255	255	433	208	170			111	170	199	374	1.81	379	199.2	17.8	342	*1	65.	332	211	2.30	255	399	25
255			255	255	255	255	252	565	110	32		64	135		179	167	140	163	177	17.4	1.59	64	42	32	123	222	251	255	25
255			258		255	255		153				64	145	174	17.2	162		128	172	177	172	89	42	76	119	162	229	255	25
255			255	255	255	250	167	109	110	97	79	115	-167	173	167	160	153	138	149	17.4	147	124	47	154	125	110	122	230	
255			255	255	255	214	41	140	128	62	42	126	173	177	17.3	165	160	164	170	17.1	105	145	97	66	102	125	01	153	25
248			255		255	170	83	145	171	90	192	152	171	176	17.2	161	157	100	143	172	17.5	156	140	102	101	3.50	07.	202	25
		155	216	253	345	17.0	118	122	100	166	1.68	100	181	179	167	154	1.50	154		169	178	172	163	167	187	135	94	160	25
255		777	207	243	235	142	125		153	173	173	188	16.7	17.3	1.67	148	1.39	147	1.54	166	182	192	182	173	182	335	79.	141	- 24
	255	273	223	231	183	142	106		134	165	1014	190	394	168	1.50	128	114	119	134	156	17.8	197	203	179	182	110	78	140	24
255			218	217	163	149	94	21.	116	187	199	186	155	149	125	127	103	100	111	134	154	143	199	195	161	100	81	117	21
244		189	313	214	17.5	141	114	78	84.	128	204	189	140	1.26	114	101	44	114	109	120	128	1.50	197	223	1.36	78.	80	103	23
	245	187	292	150	345	102	131	82	79.	145	209	174	343	122	111	100	84.	85	99	115	133	142	181	229	221	399	21	118	22
255		178	172	196	343	75	116		84	167	200	153	314	92	66	65	71	70	63	74	101	113	174	229	226	102	113	129	22
255 :			373	207	192	75	72		85	156	104	155	43	82 -	43	31	28	28		67	93	126	179	235	201	47	114	#1	: 18
255			194	235	17.5	43	53	155		145	180	149	144	40	53	35	22	24			**	17.9	175	203	170	101	78	63	10
255		215	199	236	172	70	57	77	87	131	180	142	156	100	22	36	36	43	47	19	118	189	162	204	164	65	68	50	18
255			190	229	177	72	50	61	60	114	182	154	128	1.54	81	56	56	58	69	96	185	1.57	163	221	140	.52	-09	90	21
255			203		175	103	60		54	109	149	163	143	1.37	145	101	88.	85	1046	165	150	157	195	254	200	48.	63.	150	23
200	155	123	226	236	173	140	82		45	146	217	205	180	142	150	164	146	144	161	155	184	231	253	255	250	136	60	164	25
202			147	224	174	156	105	75	69	177	249	255	247	209	166	145	123			179	234					244	106	212	
254	245	212	145	148	376	1-49	140	121	144	1.59	180	224	254	255	255	215	1.52	152			255	255	255	255	255	255	255	255	25
255.	255	255	225	169	160	125	112		1.58	159	156	160	387	229	255	232	102	94	142	230	255	255	255	255	255	255	255	255	- 25
255		255	235	222	167	110	105	50	122	166	157	161	154	161	192	209	103	43	114	172	222					255	255	255	24
255	255	255	255	253	160	114	47	84	91	142	183	229	219	199	140	154	113	.81	40	122	160	244	255	255	255	255	255	255	25
235	255	255	235	235	217	108	44	80	41	117	215	255	255	253	226	179	514	27	101	103	134	243	255	235	255	255	255	255	21
255						177		112					255	255	255			153			212					255		255	21
255	255	255	258	255	248	255	248	245	251	285	255	255	255	255	255	255	255	254	253	255	255	255	255	255	255	255	255	255	25

- Image is a matrix with integer values
- We will typically denote it with I
- *I*(*i*, *j*) is called **intensity**



- Image is a matrix with integer values
- We will typically denote it with I
- I(i,j) is called **intensity**
- Matrix I can be $m \times n$ (grayscale)





- Image is a matrix with integer values
- We will typically denote it with I
- I(i,j) is called **intensity**
- Matrix I can be $m \times n$ (grayscale)

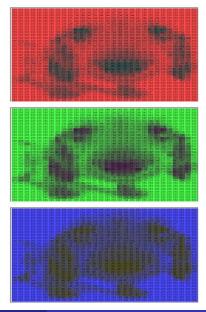
• or $m \times n \times 3$ (color)



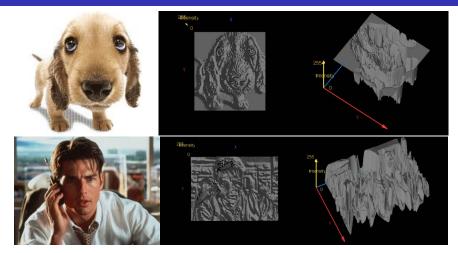
255 255							255		255	255	255	255			24P	242	241	242						255	255	255	255	25
		255				255	255	255			252						222		240				255	255	255	255	255	250
							255		253	238	237	240		228	215	210	217	227	239		243	243		255	255	255	255	25
		255			255	255	255			218				212	195	185	197	216	224	231		216	226	252	255	255	255	25
		255	255	235	255	255	255		224	206	191		215		178			200	188		372	197	210	237	255	255	255	25
255 255		522	502	530	530		254								12.0						240	125			2.50	522	520	25
255 255 255	255	220	200	230	255	252	233	206		3.10	121	151		186	174			183	17.5	161	222		1112	211	2.32	232		25
		255	200		255			151					174		167	48			177	139	100			110	162	231	255	
		255	200			100	100	118		enii	000		123					169		14	122		11.4	10.2	102	1000	220	25
255 255		255					123				124			173			164			165	852	22	122	220		132	230	125
	229				176	248	1.46	171				171	176					103		171	150	140	10.00	165		2010	202	
255 224			111		128			180			166		178	147	124	100	154	167	160	178		161	127	187	125	800	160	155
255 255					162		10.0	153		173	188	191	123	162	148			154		180	192	182		180	1115	201		24
255 255			231	185	042	106	51	136	185	174	108	184	148	150			110			175	197	203	179	182	100		140	124
255 252	258	210	217	162	140	545		116	107	100	105	155	140	125	107		000		134	154	163	100	195	361	300	610	117	22
255 244	180	213	214	171	141			04	150	206	100	140	124	1224		00	24	100	120	120	150	197		124	201	DO:		22
255 240	101	202	196	145	102		0.0	29	145	210	174	143				04	115	-00		105	142	101	230	221	104	OT.		23
255 236	170	172	196	103	251	116		84		200	153	104	92	2017	6350			6377	741		113		220	226	302		129	23
				197					156	184											124	170	211	201	87	114	550	19
255 246		194		175					145				89.0							02.1	179		263	178	201			18
		190	236	172					121	180	142	156	100							118	189	162	200	164	6.5			18
255 240		100.	229	177	22						154		134						5.0		157	163		348			1000	21
255 230		203	229	175	363				169	169				145			155	105	105	150	157	195	254		68.		150	25
	122		236	173	140		61		146									161			231	253		250	886	192	164	25
203 152		197	224	174	156		2440		177	249	255	247		166				143			255	255	255	255	244	196	212	25
254 240		165	108		0.49	140		144	159	180	224	521	255	522	215		152			255	520	5222	250	320	592	255	250	25
235 235	255	225	109	160				158	194	106	160	187.			232		28.1	142		255	255	255	255	255	222	255	255	25
255 255	255	255	253	100	88		101	122	100	157	101	356		192				114		233		255	255		255	255		23
			255		114			01	142	199	229	219	189	160	179		-	88 101			833	255	255	255	255	255	250	25
		255	244				114	147	100	6.9	255	255	253	255	202	100	100	151	1.1	212	1155		255	255		255	255	
255 255		255	222					251															255			255		

- Image is a matrix with integer values
- We will typically denote it with I
- *I*(*i*, *j*) is called **intensity**
- Matrix I can be $m \times n$ (grayscale)
- or $m \times n \times 3$ (color)





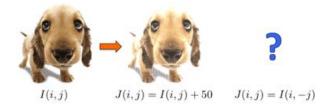
Intensity



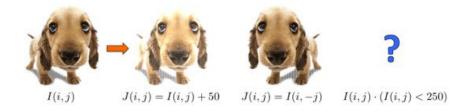
- We can think of a (grayscale) image as a function $f : \mathbb{R}^2 \to \mathbb{R}$ giving the intensity at position (i, j)
- Intensity 0 is black and 255 is white

$$\begin{array}{c} & & & \\ & & & \\ I(i,j) \end{array} \longrightarrow \begin{array}{c} & & \\ & & \\ J(i,j) = I(i,j) + 50 \end{array}$$

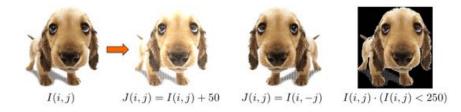
• We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)



• We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)



• We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)



• We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)

Linear Filters

Reading: Szeliski book, Chapter 3.2

Motivation: Finding Waldo

• How can we find Waldo?





[Source: R. Urtasun]

Sanja Fidler

Answer

- Slide and compare!
- In formal language: filtering

Motivation: Noise reduction

• Given a camera and a still scene, how can you reduce noise?



Image Filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel
- In other words... Filtering



Local image data

Modified image data

[Source: L. Zhang]

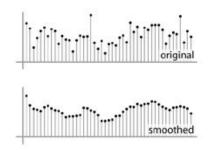
Applications of Filtering

- Enhance an image, e.g., denoise.
- Detect patterns, e.g., template matching.
- Extract information, e.g., texture, edges.
- Filtering is used in Convolutional Neural Networks

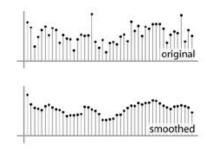
Applications of Filtering

- Enhance an image, e.g., denoise. Let's talk about this first
- Detect patterns, e.g., template matching.
- Extract information, e.g., texture, edges.

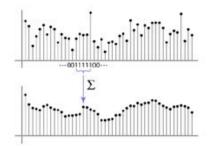
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



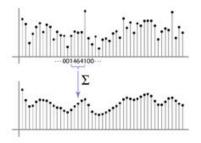
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



- Simplest thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- Moving average in 1D: [1, 1, 1, 1, 1]/5



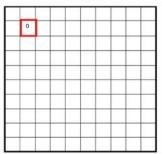
- Simplest thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- Non-uniform weights [1, 4, 6, 4, 1] / 16

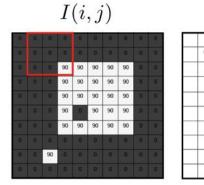


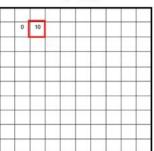
I(i, j)

G(i, j)

9					0		220		
0		0							
		0	90	90	90	90	90		
	D	0	90	90	90	90	90	Ð	
			90	90	90	90	90	0	
a			90	0	90	90	90	(0))	
			90	90	90	90	90	0	
			0	0	0	ø		0	
		90							
		0							





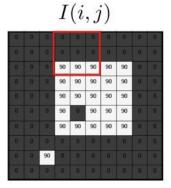


G(i,j)

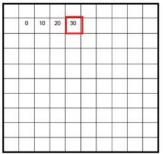
I(i, j)

G(i, j)

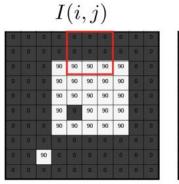
		Г	0			0	0	0	0	
10	0		0			0	0		:0	
			0	90	90	90	90	90	0	
			0	90	90	90	90	90	0	
	-		0	90	90	90	90	90		
			.0	90	90	90	0	90		
	-		.0	90	90	90	90	90		
			0	0	0	0	0	D		
			.0						90	
			.0		0				0	



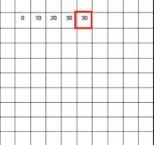




Moving Average in 2D







[Source: S. Seitz]

Moving Average in 2D

I(i, j)

G(i, j)

	00		90	90	90	90	90		
			90	90	90	90	90	10	
			90	90	90	90	90	0	
	000		90	0	90	90	90	0	
	0.		90	90	90	90	90	. 0	
0	0		0	D	G	0	0		
		90							
		0						÷01	

[Source: S. Seitz]

• Involves weighted combinations of pixels in small neighborhoods:

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u,j+v)$$

 The output pixels value is determined as a weighted sum of input pixel values

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u,j+v)$$

• Involves weighted combinations of pixels in small neighborhoods:

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u,j+v)$$

 The output pixels value is determined as a weighted sum of input pixel values

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u,j+v)$$

• The entries of the weight kernel or mask F(u, v) are often called the filter coefficients.

• Involves weighted combinations of pixels in small neighborhoods:

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u,j+v)$$

 The output pixels value is determined as a weighted sum of input pixel values

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u,j+v)$$

- The entries of the weight kernel or mask F(u, v) are often called the filter coefficients.
- This operator is the correlation operator

$$G = F \otimes I$$

• Involves weighted combinations of pixels in small neighborhoods:

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u,j+v)$$

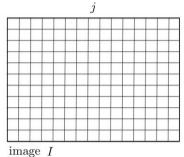
 The output pixels value is determined as a weighted sum of input pixel values

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u,j+v)$$

- The entries of the weight kernel or mask F(u, v) are often called the filter coefficients.
- This operator is the correlation operator

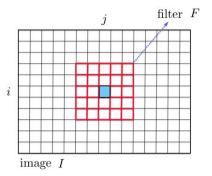
$$G = F \otimes I$$

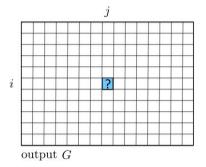
• It's really easy!



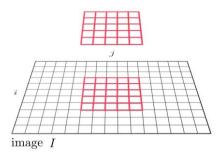
filter F

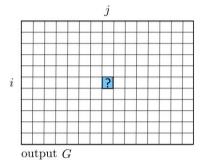
• It's really easy!



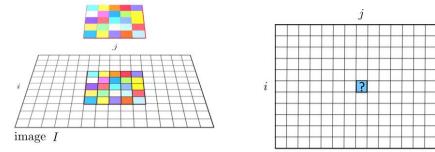


• It's really easy!





• It's really easy!

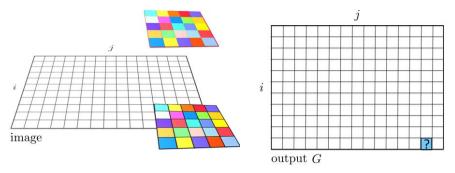


output G

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u,j+v)$$

 $G(i,j) = F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + \dots + F(\square) \cdot I(\square)$

• What happens along the borders of the image?



$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u,j+v)$$

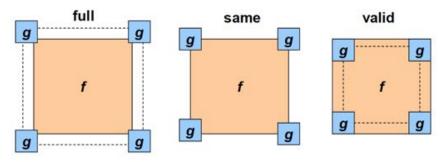
 $G(i,j) = F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + \ldots + F(\square) \cdot I(\square)$

Boundary Effects

- What happens at the border of the image? What's the size of the output matrix?
- MATLAB: FILTER2(G, F, SHAPE) Python: SCIPY.NDIMAGE.CONVOLVE
- shape = "full" output size is sum of sizes of f and g
- shape = "same": output size is same as f
- shape = "valid": output size is difference of sizes of f and g

Boundary Effects

- What happens at the border of the image? What's the size of the output matrix?
- MATLAB: FILTER2(G, F, SHAPE) Python: SCIPY.NDIMAGE.CONVOLVE
- shape = "full" output size is sum of sizes of f and g
- shape = "same": output size is same as f
- shape = "valid": output size is difference of sizes of f and g



• What's the result?





Original

• What's the result?







Filtered (no change)

Original

• What's the result?





Original

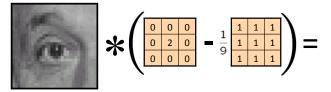
• What's the result?





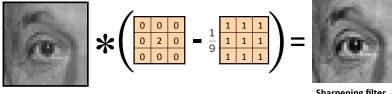


• What's the result?



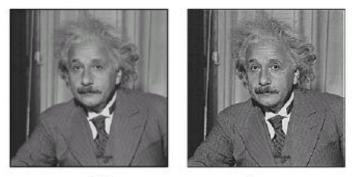
Original

• What's the result?



Original

Sharpening filter (accentuates edges)



before

after

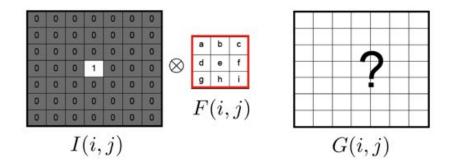
Sharpening



[Source: N. Snavely]

Example of Correlation

 What is the result of filtering the impulse signal (image) I with the arbitrary filter F?



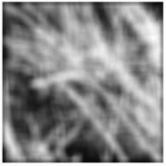
Smoothing by averaging



depicts box filter: white = high value, black = low value



original



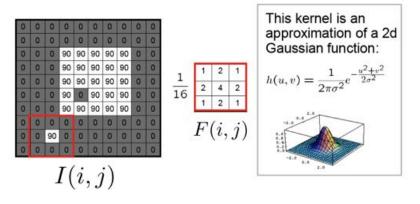
filtered

• What if the filter size was 5 x 5 instead of 3 x 3? [Source: K. Graumann]

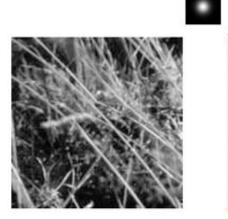
Sanja Fidler

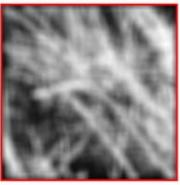
Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?
- Removes high-frequency components from the image (low-pass filter).

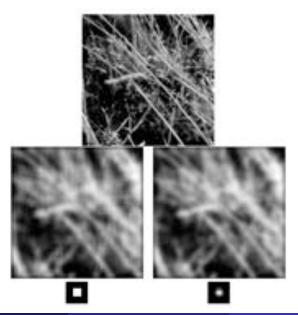


Smoothing with a Gaussian

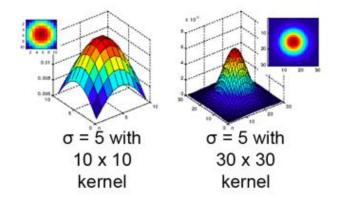




Mean vs Gaussian

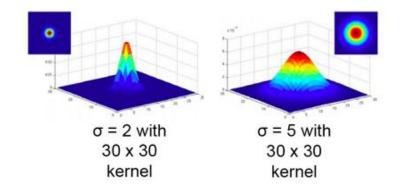


• Size of filter or mask: Gaussian function has infinite support, but discrete filters use finite kernels.

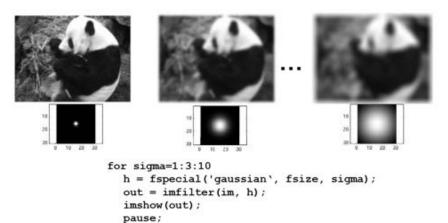


Gaussian filter: Parameters

• Variance of the Gaussian: determines extent of smoothing.



Gaussian filter: Parameters

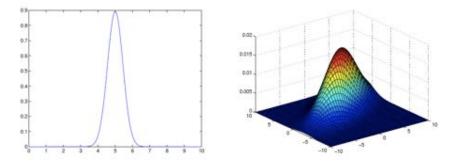


end

Is this the most general Gaussian?

• No, the most general form for $\mathbf{x} \in \Re^d$

$$\mathcal{N}\left(\mathbf{x};\,\mu,\Sigma
ight)=rac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left(-rac{1}{2}(\mathbf{x}-\mu)^{\mathcal{T}}\Sigma^{-1}(\mathbf{x}-\mu)
ight)$$



• We typically use isotropic filters (i.e., circularly symmetric)

Sanja Fidler

Properties of the Smoothing

- All values are positive.
- They all sum to 1.

Properties of the Smoothing

- All values are positive.
- They all sum to 1.
- Amount of smoothing proportional to mask size.

Properties of the Smoothing

- All values are positive.
- They all sum to 1.
- Amount of smoothing proportional to mask size.
- Remove high-frequency components; low-pass filter.

Note: This holds for smoothing filters, not general filters

- All values are positive.
- They all sum to 1.
- Amount of smoothing proportional to mask size.
- Remove high-frequency components; low-pass filter.

Note: This holds for smoothing filters, not general filters

Finding Waldo





image I

• How can we use what we just learned about filtering to find Waldo?

Finding Waldo



image I



filter F

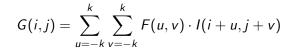
• Is correlation a good choice?

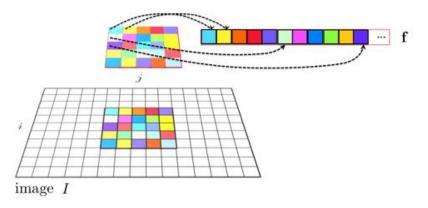
• Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u,j+v)$$

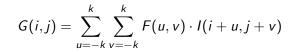
• Can we write that in a more compact form (with vectors)?

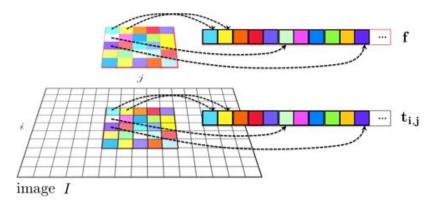
• Remember correlation:



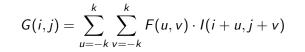


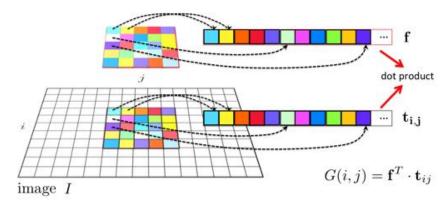
• Remember correlation:





• Remember correlation:





• Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u,j+v)$$

• Can we write that in a more compact form (with vectors)?

• Define
$$\mathbf{f} = F(:)$$
, $T_{ij} = I(i - k : i + k, j - k : j + k)$, and $\mathbf{t}_{ij} = T_{ij}(:)$

$$G(i, j) = \mathbf{f}^T \cdot \mathbf{t}_{ij}$$

where \cdot is a dot product

• Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u,j+v)$$

• Can we write that in a more compact form (with vectors)?

• Define
$$\mathbf{f} = F(:)$$
, $T_{ij} = I(i - k : i + k, j - k : j + k)$, and $\mathbf{t}_{ij} = T_{ij}(:)$
$$G(i,j) = \mathbf{f}^T \cdot \mathbf{t}_{ij}$$

where \cdot is a dot product

• Homework: Can we write full correlation $G = F \otimes I$ in matrix form?

• Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u,j+v)$$

• Can we write that in a more compact form (with vectors)?

• Define
$$\mathbf{f} = F(:)$$
, $T_{ij} = I(i - k : i + k, j - k : j + k)$, and $\mathbf{t}_{ij} = T_{ij}(:)$
$$G(i,j) = \mathbf{f}^T \cdot \mathbf{t}_{ij}$$

where \cdot is a dot product

• Finding Waldo: How could we ensure to get the best "score" (e.g. 1) for an image crop that looks exactly like our filter?

• Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u,j+v)$$

• Can we write that in a more compact form (with vectors)?

• Define $\mathbf{f} = F(:)$, $T_{ij} = I(i - k : i + k, j - k : j + k)$, and $\mathbf{t}_{ij} = T_{ij}(:)$

$$G(i,j) = \mathbf{f}^T \cdot \mathbf{t}_{ij}$$

where \cdot is a dot product

- Finding Waldo: How could we ensure to get the best "score" (e.g. 1) for an image crop that looks exactly like our filter?
- Normalized cross-correlation:

$$G(i,j) = rac{\mathbf{f}^T \cdot \mathbf{t}_{ij}}{||\mathbf{f}|| \cdot ||\mathbf{t}_{ij}||}$$

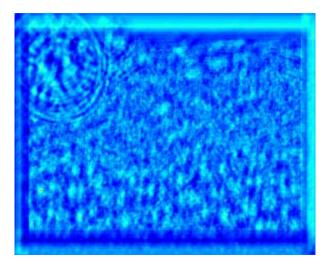
Back to Waldo



image I

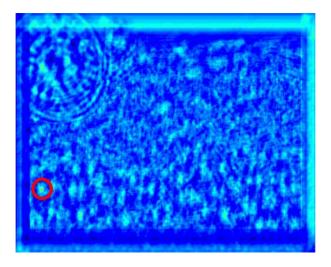






• Result of normalized cross-correlation

Sanja Fidler



• Find the highest peak

Sanja Fidler

Back to Waldo



And put a bounding box (rectangle the size of the template) at the point!

Back to Waldo



• Homework: Do it yourself! Code on class webpage. Don't cheat!

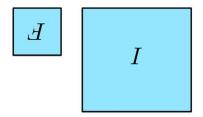
• Convolution operator

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i-u,j-v)$$

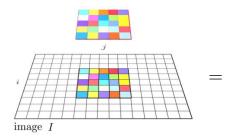
• Convolution operator

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i-u,j-v)$$

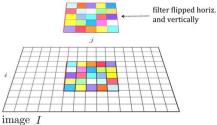
• **Equivalent** to flipping the filter in both dimensions (bottom to top, right to left) and apply correlation.



Correlation vs Convolution



Correlation



Convolution

• For a Gaussian or box filter, how will the outputs F * I and $F \otimes I$ differ?

Correlation vs Convolution

- For a Gaussian or box filter, how will the outputs F * I and $F \otimes I$ differ?
- How will the outputs differ for:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Correlation vs Convolution

- For a Gaussian or box filter, how will the outputs F * I and $F \otimes I$ differ?
- How will the outputs differ for:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

• If the input is an impulse signal, how will the outputs differ? $\delta * I$ and $\delta \otimes I$?

"Optical" Convolution

• Camera Shake



Figure: Fergus, et al., SIGGRAPH 2006

• Blur in out-of-focus regions of an image.



Figure: Bokeh: http://lullaby.homepage.dk/diy-camera/bokeh.html Click for more info [Source: N. Snavely]

Sanja Fidler

Properties of Convolution

Commutative :
$$f * g = g * f$$

Associative : $f * (g * h) = (f * g) * h$
Distributive : $f * (g + h) = f * g + f * h$
Assoc. with scalar multiplier : $\lambda \cdot (f * g) = (\lambda \cdot f) * h$

Properties of Convolution

Commutative : f * g = g * fAssociative : f * (g * h) = (f * g) * hDistributive : f * (g + h) = f * g + f * hAssoc. with scalar multiplier : $\lambda \cdot (f * g) = (\lambda \cdot f) * h$

• The Fourier transform of two convolved images is the product of their individual Fourier transforms:

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

Properties of Convolution

Commutative : f * g = g * fAssociative : f * (g * h) = (f * g) * hDistributive : f * (g + h) = f * g + f * hAssoc. with scalar multiplier : $\lambda \cdot (f * g) = (\lambda \cdot f) * h$

• The Fourier transform of two convolved images is the product of their individual Fourier transforms:

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

- **Homework:** Why is this good news?
- Hint: Think of complexity of convolution and Fourier Transform
- Both correlation and convolution are linear shift-invariant (LSI) operators: the effect of the operator is the same everywhere.

• Convolving twice with Gaussian kernel of width σ is the same as convolving once with kernel of width $\sigma\sqrt{2}$



• We don't need to filter twice, just once with a bigger kernel

[Source: K. Grauman]

• The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?
- In many cases (**not all!**), this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only** 2*K* **operations**.

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?
- In many cases (**not all!**), this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only** 2*K* **operations**.
- If this is possible, then the convolution filter is called **separable**.

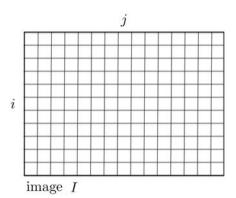
- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?
- In many cases (**not all!**), this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only** 2*K* **operations**.
- If this is possible, then the convolution filter is called **separable**.
- And it is the outer product of two filters:

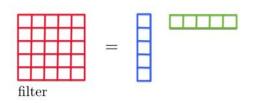
$$\mathbf{F} = \mathbf{v} \mathbf{h}^T$$

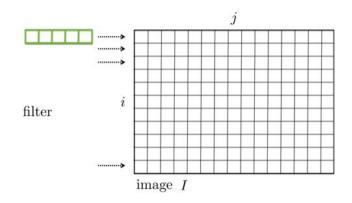
• Homework: Think why in the case of separable filters 2D convolution is the same as two 1D convolutions

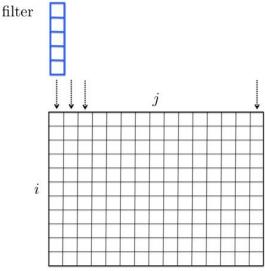
[Source: R. Urtasun]









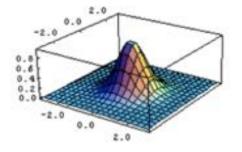


output of horizontal convolution

Separable Filters: Gaussian filters

• One famous separable filter we already know:

Gaussian :
$$f(x,y) = \frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{\sigma^2}}$$

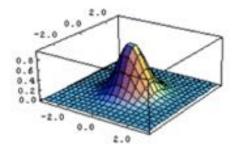


Separable Filters: Gaussian filters

• One famous separable filter we already know:

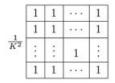
Gaussian :
$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}$$

= $\left(\frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{x^2}{\sigma^2}}\right) \cdot \left(\frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{y^2}{\sigma^2}}\right)$



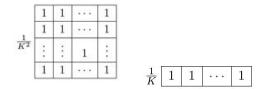
Let's play a game...

Is this separable? If yes, what's the separable version?



[Source: R. Urtasun]

Is this separable? If yes, what's the separable version?

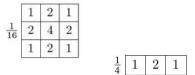


What does this filter do?

Is this separable? If yes, what's the separable version?

$\frac{1}{16}$	1	2	1
	2	4	2
	1	2	1

Is this separable? If yes, what's the separable version?

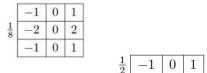


What does this filter do?

Is this separable? If yes, what's the separable version?

	-1	0	1
$\frac{1}{8}$	-2	0	2
Ĩ	-1	0	1

Is this separable? If yes, what's the separable version?



What does this filter do?

• Inspection... this is what we were doing.

- Inspection... this is what we were doing.
- Looking at the analytic form of it.

- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U}\Sigma\mathbf{V}^{T} = \sum_{i} \sigma_{i} u_{i} v_{i}^{T}$$

with $\Sigma = \operatorname{diag}(\sigma_i)$.

- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U} \Sigma \mathbf{V}^{T} = \sum_{i} \sigma_{i} u_{i} v_{i}^{T}$$

with $\Sigma = \text{diag}(\sigma_i)$.

 $\bullet \ \mbox{Matlab:} \ \ [U,S,V] = {\rm SVD}(F);$

- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U} \Sigma \mathbf{V}^{T} = \sum_{i} \sigma_{i} u_{i} v_{i}^{T}$$

with $\Sigma = \operatorname{diag}(\sigma_i)$.

- $\bullet \ \mbox{Matlab:} \ \ [U,S,V] = {\rm SVD}(F);$
- $\sqrt{\sigma_1}\mathbf{u}_1$ and $\sqrt{\sigma_1}\mathbf{v}_1^T$ are the vertical and horizontal filter.

Summary – Stuff You Should Know

- Correlation: Slide a filter across image and compare (via dot product)
- Convolution: Flip the filter to the right and down and do correlation
- Smooth image with a Gaussian kernel: bigger σ means more blurring
- **Some** filters (like Gaussian) are **separable**: you can filter faster. First apply 1D convolution to each row, followed by another 1D conv. to each column
- Applying first a Gaussian filter with σ_1 and then another Gaussian with σ_2 is the same as applying one Gaussian filter with $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$

Functions

Python functions:

- SCIPY.NDIMAGE.CORRELATE: correlation
- SCIPY.NDIMAGE.CONVOLVE: convolution
- Many filters available: https://docs.scipy.org/doc/scipy-0.15.1/ reference/ndimage.html#module-scipy.ndimage.filters

Matlab functions:

- IMFILTER: can do both correlation and convolution
- CORR2, FILTER2: correlation, NORMXCORR2 normalized correlation
- CONV2: does convolution
- FSPECIAL: creates special filters including a Gaussian



• What does blurring take away?







[Source: S. Lazebnik]

Next time: Edge Detection