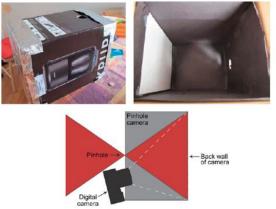
Cameras and Images

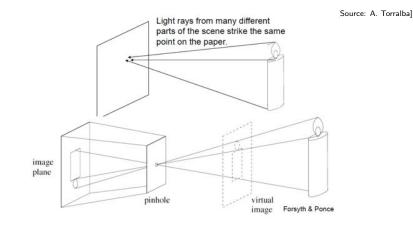
Pinhole Camera



[Source: A. Torralba]

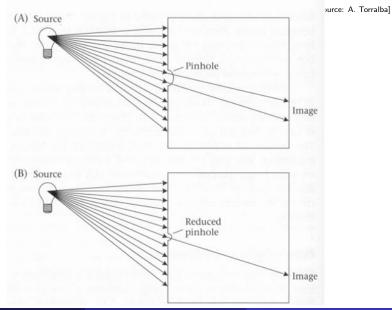
- Make your own camera
- http://www.foundphotography.com/PhotoThoughts/archives/2005/ 04/pinhole_camera_2.html

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



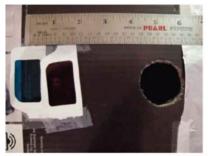
Pinhole Camera – Example

[Source: A. Torralba]





[Source: A. Torralba]



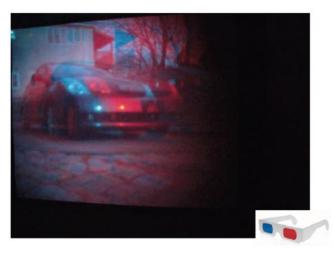


• You can make it stereo

Sanja Fidler

Pinhole Camera – Stereo Example

[Source: A. Torralba]

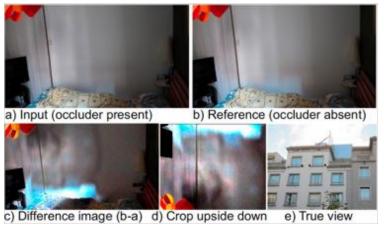


• Try it with 3D glasses!

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

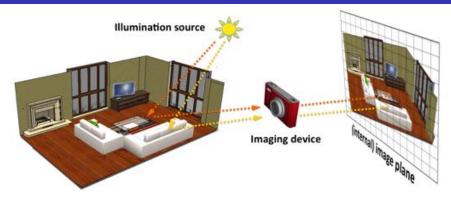


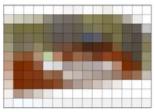
Image formation process producing a particular image depends on:

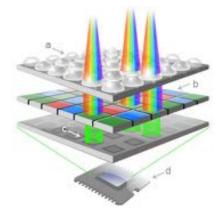
- lighting conditions
- scene geometry
- surface properties
- camera optics

Continuous image projected to sensor array



Sampling and quantization





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

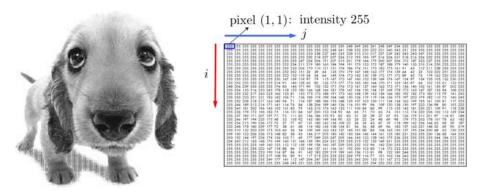
- Sample the 2D space on a regular grid
- Quantize each sample (round to nearest integer)

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



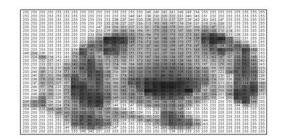
285 2																										255	255	255	25
255 2							255															251				255	255	255	25
255 3	255	255	255	255	255	255	255	255	255	253				235	228	718	210	217		239		243	243	247	255	255	255	255	21
284.3	255.	255	255	255	235	255	255	-255	255	232	210	227	227	226	212	143	185	187	214	224	231	218	216	236	252	255	255	255	. 71
255 2	255	255	255	255	255	255	255	235	247	224	204	191	207	215	201	178	164	179	200	504	294	172	187	223	237	255	255	255	23
255 2		255	255	255	255	255	255	254	211	219	100	160	364	194	191	170	153	172	147	100	179	140	155	230	214	239	255	255	25
286.2		255	255	255	255	255	255	333	206	170	110	121	151	174	106	374	1.51	170	183	178	141	91.	65.	112	211	239	255	255	31
255.2				255	255	255	252	292	151	32		64		170	179	167	1-80	163	177	17.4	159	64	42	32	123	222	251	255	28
255 1		255			255	255	223	153	119			64	149	174	17.2	142	1:50	159	172	177	172	89	42	76	119	142	220	255	2
255 2	255	255	255	255	255	2.50	167	-109	114	97	79	115	167	173	167	360	153	158	149	174	-147	124	47	354	125	184	1.22	220	2
255 3	255	255	255	255	255	214	91	140	128	62	82	126	175	177	173	165	160	164	100	171	165	145	97	66	102	125	61	153	22
248.2			255		255	176	03	145	171	90	102	152	373	176	173	161	157	160	143	172	171	156	140	102	161	150	87	202	2
255 2					345	17.0	118	122	100	166	148	166	181	179	167	154	150	154		169	178	173	163	167	187	135	94	168	12
255 2		232		243	225	142	125		1.53	173	173	188		173	1.62	148	1:39	143	154	166	182	192	182	173	182	115	79.		- 24
	255	233	221	231	183	142	106	24	1.36	165	1014	190	384	168	1.50	128	119	119	134	156	125	197	203	179	182	150	75	140	24
255 2	2.52	208	218	217	143	149	9.6	71	114	187	100	186	155	140	125	157	103	100	222	134	154	143	199	195	161	100	81	117	2
245 2	244	189	213	214	17.1	141	114	74	84	158	204	194	140	136	114	101	- 99	64	109	120	158	1.50	197	223	136	-10	80	105	2
255 2	245	1812	202	140.6	145	102	131	83	70.	145	210	174	143	133	111	100	54	85	00	115	133	142	181	230	221	308	01	118	2
255 2	236	178	172	196	163	75	116	101	64	167	200	153	104	92	66	65			63	74	101	113	174	229	226	102	113	129	2
255 3	237	160	37.5	207	197	71	72	111	85	154	104	155	43	82	43	31	28	28		67	93	126	179	235	201	47	114	\$1.	: 14
288 2	244	197	104	235	17.5	45	53	105	92	145	180	149	144	60	53	24	22	24	40	60	44	179	175	203	170	344	78	63	1.1
255 3	254	215	190	236	172	70	57	77	87	131	180	142	154	100	22	36	36	43	47	19	118	189	162	204	164	65	68	50	. 11
255 3	548	178	190	229	177	72	58	61	60	114	182	154	128	154	81	58	56	58	69	96	165	157	163	223	140	62	49	96	21
255 3	210	132	203	229	175	103	60	54	54	109	149	163	143	1:27	145	101	88	85	104	165	1.50	157	195	254	200	48	63	150	2
200 1	155	123	226	236	173	1.60	82	58	45	146	217	205	180	142	1.50	164	144	144	141	155	180	231	253	258	250	134	68	164	24
202 1	152	144	1977	224	174	156	105	25	69	177	249	255	247	209	166	145	123	129	143	179	234	255	255	255	255	244	104	212	2
254 2	345	212	145	148	176	1-89	140	121	144	1.58	180	224	255	255	255	215	1.52	152	149	241	355	255	255	255	255	255	255	255	2
265	255	255	225	149	140	125	112	112	1.54	1.54	154	140	187	229	255	222	102	-	142	236	255	255	235	255	255	255	255	255	- 51
244.0	255	255	248	222	167	110	-	64	122	166	157	161	154	161	192	200	103	83	114	172	222	255	255	245	255	255	255	255	2
255	355	255	255	252	160	114	47	84	91	142	180	229	110	109	140	154	115	81	60	199	360		255			255	255	255	2
144				255	217	108	14	80	41	117	215	265	258	253	224	179	114	77	101	103	154	243		255		255	255	255	- 51
266					249		141	Tip		204	247	566	255	255	245			153			212			255		264	366	544	3
255 1							248			288																	255		

- 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)

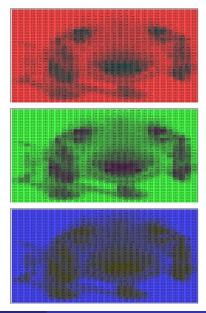
• or $m \times n \times 3$ (color)



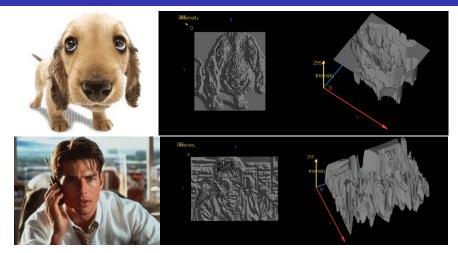
255 255							255																		255	255	255	
255 255		255				255	255	255			252		241	238	232	220	222	231	240			255		255	255	255	255	21
255 255								255		238	237	240		228	215	210	217	227	239	242		243		255	255	255	255	
255 255		255	255				255										197		224		218	216		252	255	255		21
255 255		255	255	255	255	255	255					207	215				179			206	372	197	223		255	255	255	
255 255		255	255	255	255	255	254		219	180							172		188		240				2.59	255	255	
255 255	255	255	255	285	255	255	233	206	.820				174	186			\$70		17.5		840					235	255	
255 255					255			151					170		167	148	163	177	174	:159	040			123	222	251		21
255 255 255 255		255	255		255			119		2.2	6.0								174	172	100			1128	0.62	229		21
235 255				255			1108						173								885	82	1.54			132	230	
235 235 248 208		255			214		140	120			126		177				164			165	150	87.0		102		833	1537	
					178	100	1.43	180			100		176	173	154	322				178	520	163	1996	101	150	1.200	202	
255 224 255 255				245	128		122	120	100	100	188	191	178		148		154		160	178	82	10.2	167	122	123	223	141	
255 255			24.3	223	042			133	144	82	108	184	148	150			110		100	102	122	203	120	122				
255 252			217	163	1000			116	167	100	105						100		100	17.5	122.	203	22.5	1000	222			
255 244	200			10.4	144	100		110	150		100		126	122				109	1.34	124	150	144	223	201	200			
255 240			196	100	1000				145	210		145					88	104			120		230	경험	104			2
255 236			104	103	100	214		80			153	101.4	02							103	111		220	226	100			
		171		107		-			156	184	155	03	848			-	-			81	124	170		201	225		-	ñ
255 246		104		175					145		189	146								00.	170	175		178	564	200	-	15
255 254		200		172					121		142		100								100	142	206	164				16
255 248		100	200	177				-	114	102	154	120	134	1.1					1000	183	157	163		340				2
255 230	832	203	229	175				211	100	169	103	143	127	345						150	157	195	254	200	22			2
209 1850			236	173	140				146	217				150	144	140	124		155					250	157	100	144	
203 152	144	197	224	174	156				177	249		247		166			328	143	179	236	255	255	255	255	224	186		12
254 240	212	165	TAR	170	140	140			1:50	180	224	251	255	255	215			169	241	255	255	255	255	255	255	255	255	2
255 255	255	225	309	160	125			158	1.59	156	160	187		255	232	102	94.0	142	230	255	255	255	255	255	255	255	255	2
255 255	255	255		167	3.12	88:	06-	122	166	157	161	156	161	192	200			254			255	255	255	255	255	255	255	2
255 255		255			114	87		911	1142	183	229	219	189.	100	156		RI.	880	122	180	244	255	255	255	255	255	255	2
255 255				217		10	80.	91	\$37	215		255	250	226	179	116	22		103	156	243	255		255	255	255	255	
235 255		255						147			255	255	255	255	245	200	383	151	167		255	255		255	255	255		2
255 255	255	255	255	255	255	248	245	251	255	255	255	255	255	255	255	255	254	253	255	255	255	255	255	255	255	255	255	2

- 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)





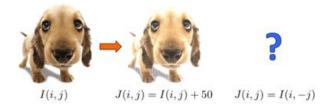
Intensity



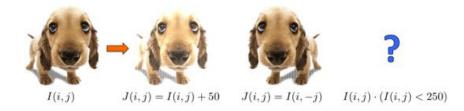
- We can think of a (grayscale) image as a function $f : \mathbb{R}^2 \to \mathbb{R}$ giving the intensity at position (i, j)
- Intensity 0 is black and 255 is white

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

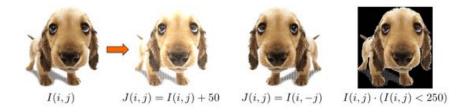
• We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)



• 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]

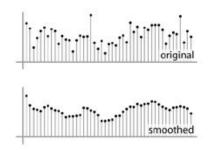
Applications of Filtering

- 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

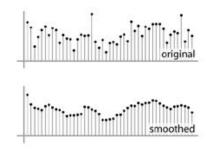
Applications of Filtering

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

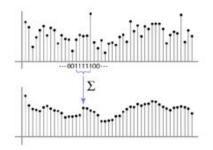
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



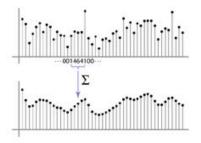
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



- 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



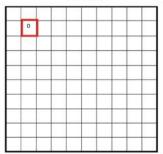
- Simplest thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- Non-uniform weights [1, 4, 6, 4, 1] / 16

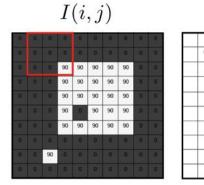


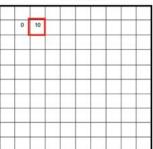
I(i, j)

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							







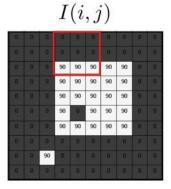
G(i,j)

I(i, j)

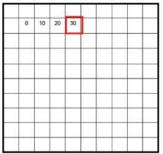
G(i,j)

	0	0	0	0				0	Г
			0					0	
		90	90	90	90	90		0	
	0	90	90	90	90	90	0	0	
		90	90	90	90	90		0	
	0	90	0	90	90	90		.0	
		90	90	90	90	90	0	0	
		D	0	0	0	0		0	
	90							.0	
	0							0	

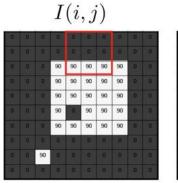
		-				
0	10	20				
			1			
			1			
			1			



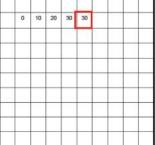




Moving Average in 2D







[Source: S. Seitz]

Moving Average in 2D

I(i, j)

G(i,j)

		90	90	90	90	90	0.0	
		90	90	90	90	90	0	
		90	90	90	90	90	0	
		90	0	90	90	90	0	
		90	90	90	90	90	0.	
0		0	D	0	0	0	0.1	0
	90	0						
	0	0					000	

[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:

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

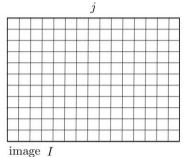
 The output pixels value is determined as a weighted sum of input pixel values

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

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

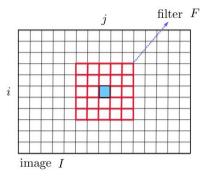
$$G = F \otimes I$$

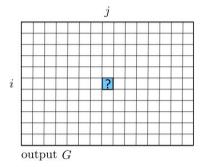
• It's really easy!



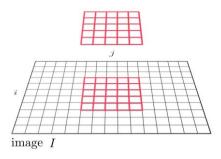
filter F

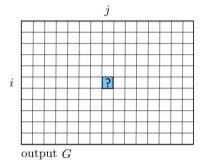
• It's really easy!



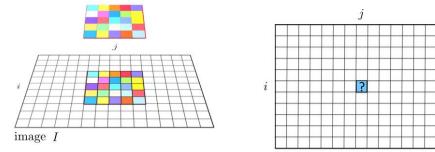


• It's really easy!





• It's really easy!

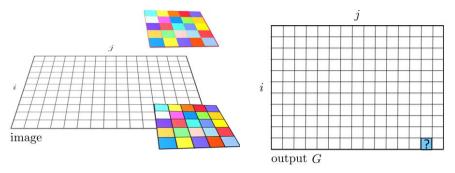


output G

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

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

• What happens along the borders of the image?



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

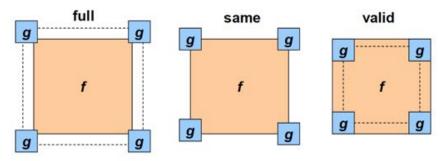
 $G(i,j) = F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + \ldots + F(\square) \cdot I(\square)$

Boundary Effects

- What happens at the border of the image? What's the size of the output matrix?
- MATLAB: FILTER2(G, F, SHAPE)
- 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)
- 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



[Source: S. Lazebnik]

Sanja Fidler

• What's the result?





Original

• What's the result?







Filtered (no change)

Original

• What's the result?





Original

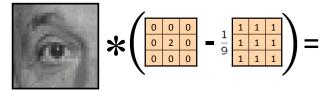
• What's the result?





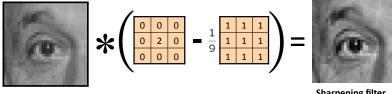


• What's the result?



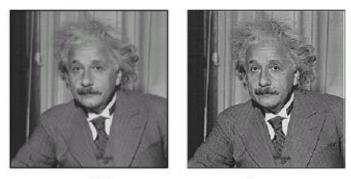
Original

• What's the result?



Original

Sharpening filter (accentuates edges)



before

after

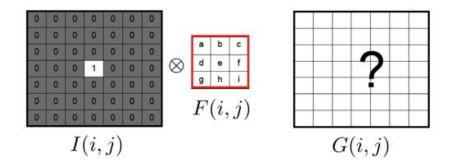
Sharpening



[Source: N. Snavely]

Example of Correlation

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



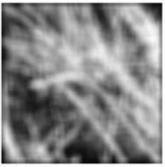
Smoothing by averaging



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



original



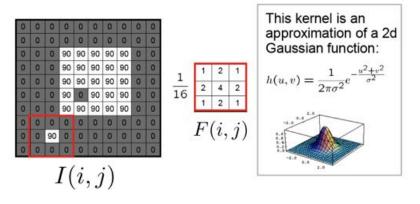
filtered

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

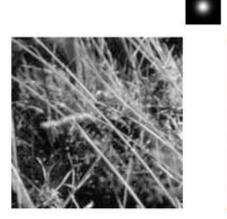
Sanja Fidler

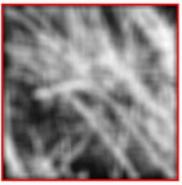
Gaussian filter

- 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).

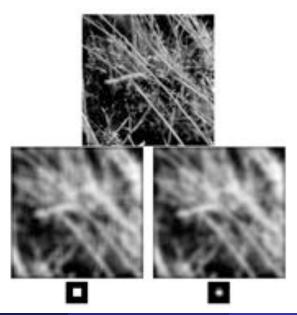


Smoothing with a Gaussian

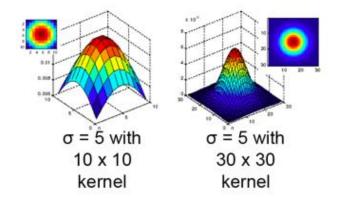




Mean vs Gaussian

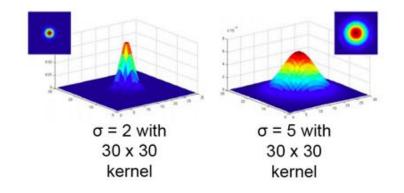


• Size of filter or mask: Gaussian function has infinite support, but discrete filters use finite kernels.

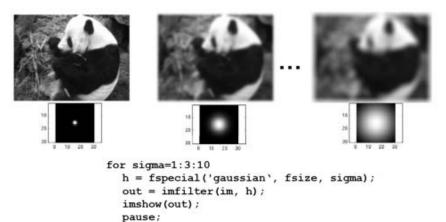


Gaussian filter: Parameters

• Variance of the Gaussian: determines extent of smoothing.



Gaussian filter: Parameters

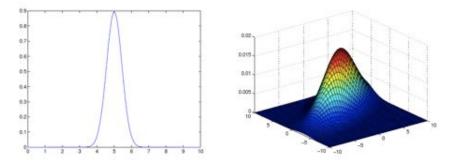


end

Is this the most general Gaussian?

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

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



• But the simplified version is typically used for filtering.

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.

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

Finding Waldo





image I

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

Finding Waldo



image *I*



filter F

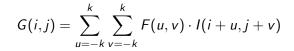
• Is correlation a good choice?

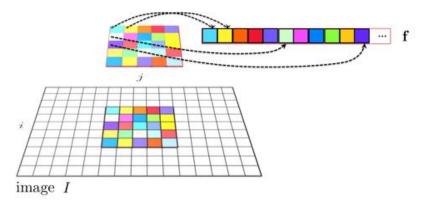
• Remember correlation:

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

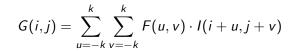
• Can we write that in a more compact form (with vectors)?

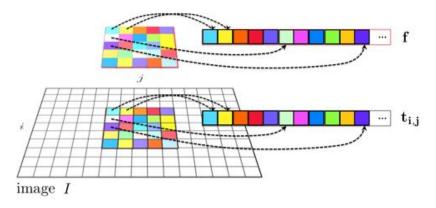
• Remember correlation:



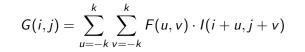


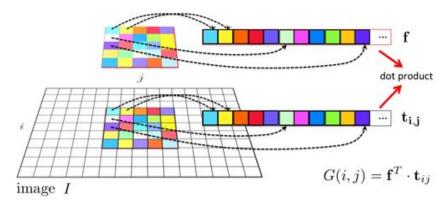
• Remember correlation:





• 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:

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

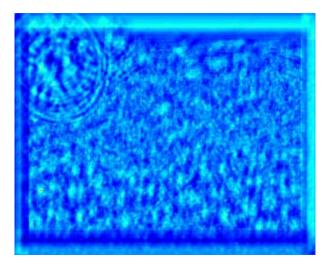
Back to Waldo



image I

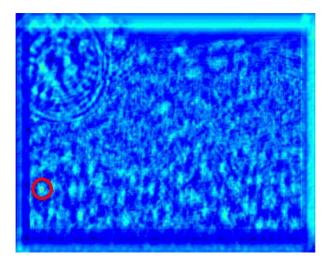






• Result of normalized cross-correlation

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 ;)

Sanja Fidler

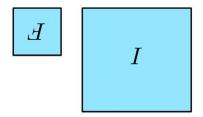
• Convolution operator

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

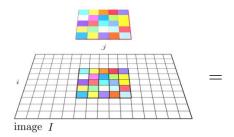
• Convolution operator

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

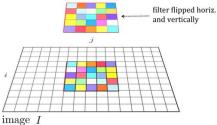
• **Equivalent** to flipping the filter in both dimensions (bottom to top, right to left) and apply correlation.



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**.

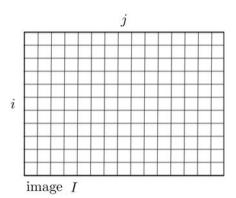
- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?
- In many cases (**not all!**), this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution,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

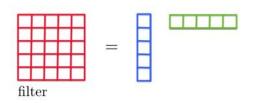
$$\mathbf{F} = \mathbf{v} \mathbf{h}^T$$

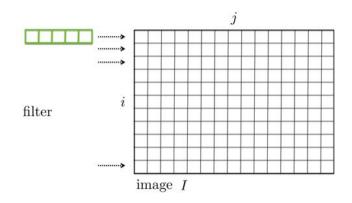
• Homework: Think why in the case of separable filters 2D convolution is the same as two 1D convolutions

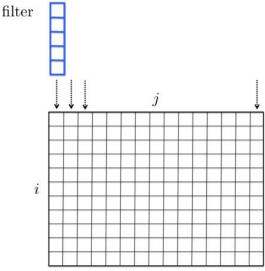
[Source: R. Urtasun]









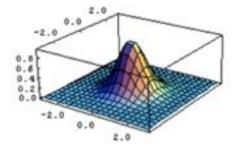


output of horizontal convolution

Separable Filters: Gaussian filters

• One famous separable filter we already know:

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

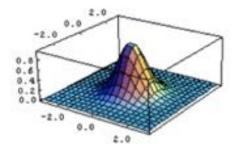


Separable Filters: Gaussian filters

• One famous separable filter we already know:

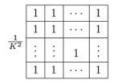
Gaussian :
$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}$$

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



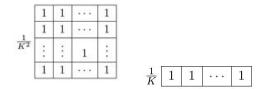
Let's play a game...

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



[Source: R. Urtasun]

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

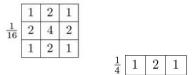


What does this filter do?

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?

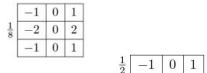


What does this filter do?

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 $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$

Matlab functions:

- IMFILTER: can do both correlation and convolution
- \bullet $_{\rm CORR2}, \, {\rm FILTER2:}$ correlation, ${\rm NORMXCORR2}$ normalized correlation
- CONV2: does convolution
- FSPECIAL: creates special filters including a Gaussian



• What does blurring take away?







[Source: S. Lazebnik]

Next time: Edge Detection