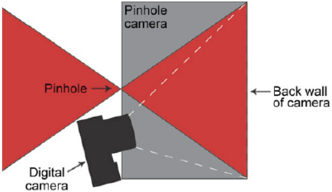
Cameras and Images

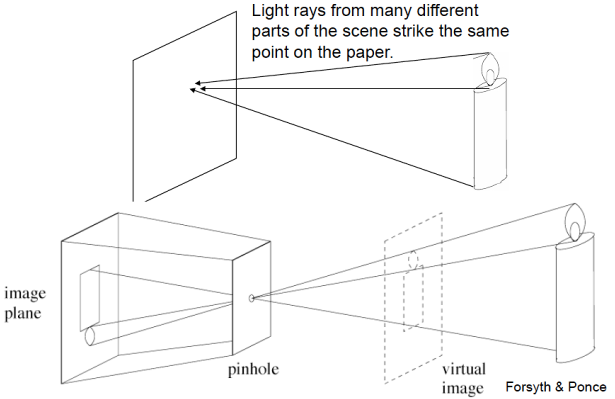
Pinhole Camera

[Source: A. Torralba]



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

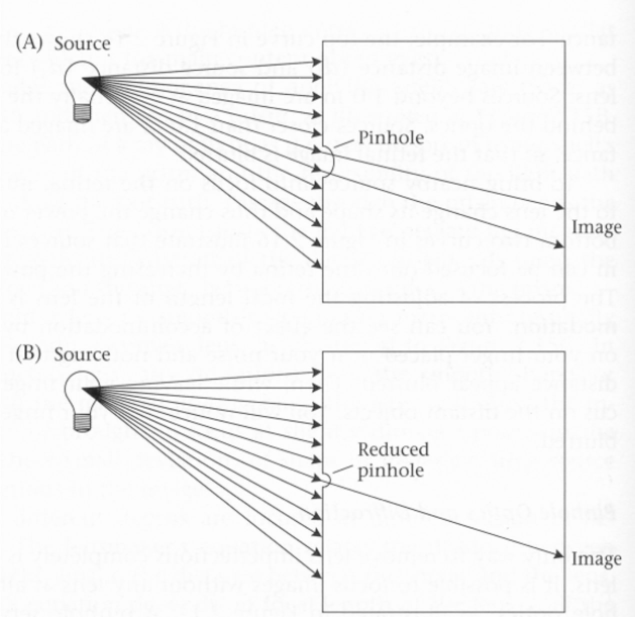
. Torralba]



- How it works
- The pinhole camera only allows rays from one point in the scene to strike each point of the paper.

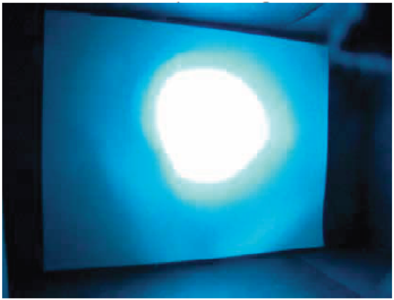
Pinhole Camera

. Torralba]



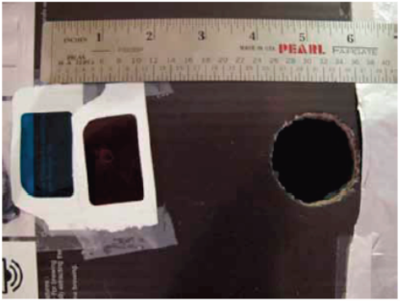
Pinhole Camera

[Source: A. Torralba]



- Example

[Source: A. Torralba]



- Example

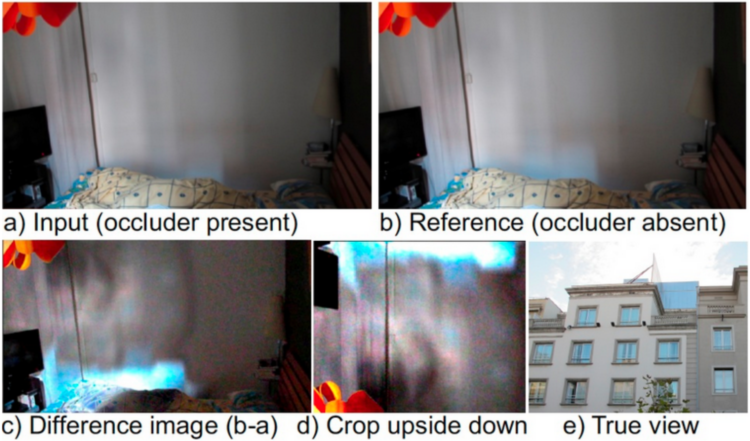
Pinhole Camera

[Source: A. Torralba]

- Example

Pinhole Camera

[Source: A. Torralba]



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

Digital Camera

[Adopted from S. Seitz]

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

Image Formation

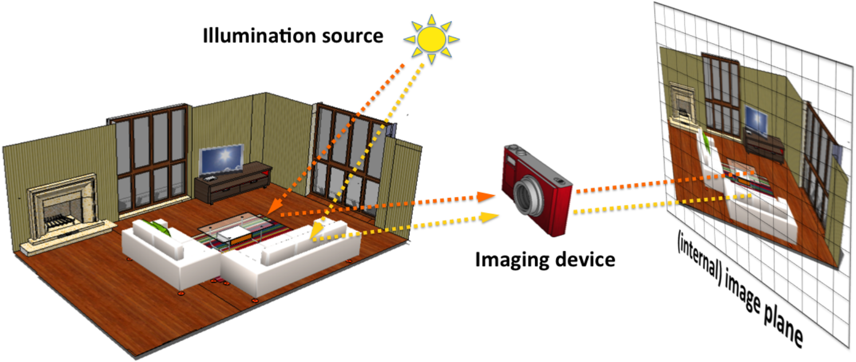
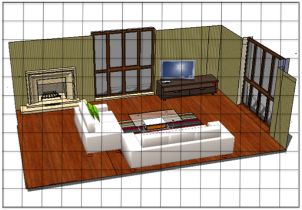


Image formation process producing a particular image depends on:

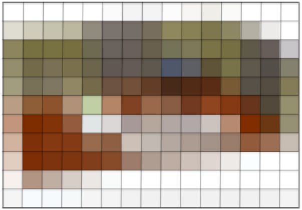
- lighting conditions
- scene geometry
- surface properties
- camera optics

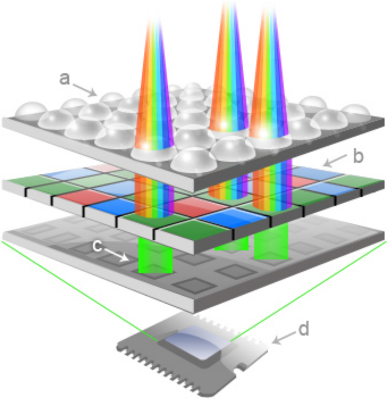
Digital Image

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)

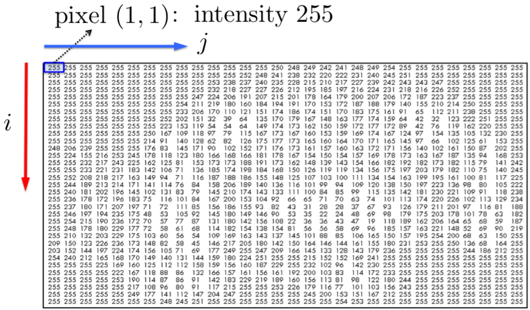
Digital Image

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

255	255	255	255	255	255	255	255	255	255	255	255	255	255	250	248	249	242	241	248	249	254	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	252	248	241	238	232	220	222	231	240	245	251	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	253	238	237	240	235	228	215	210	217	227	239	242	243	243	247	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	232	218	227	227	226	212	195	185	197	216	224	231	218	216	226	252	255	255	255	255
255	255	255	255	255	255	255	255	255	247	224	206	191	207	215	201	178	164	179	200	207	206	172	187	223	237	255	255	255	255
255	255	255	255	255	255	255	255	254	211	219	180	160	184	194	191	170	153	172	187	188	179	140	155	210	214	250	255	255	255
255	255	255	255	255	255	255	255	233	206	170	110	121	151	174	186	174	151	170	183	175	161	91	65	112	211	238	255	255	255
255	255	255	255	255	255	255	252	202	151	32	39	64	135	170	179	167	148	163	177	174	159	64	42	32	123	222	251	255	255
255	255	255	255	255	255	255	223	153	119	54	54	64	149	174	173	162	150	159	172	177	172	89	42	76	119	162	220	255	255
255	255	255	255	255	255	250	167	109	118	97	79	115	167	173	167	160	153	159	169	174	167	124	97	154	135	105	132	230	255
255	255	255	255	255	255	214	91	140	128	62	82	126	175	177	173	165	160	164	170	171	165	145	97	66	102	125	61	153	255
248	206	239	255	255	255	176	83	145	171	90	102	152	171	176	173	161	157	160	163	172	171	156	140	102	161	150	87	202	255
255	224	155	216	253	245	178	118	123	180	166	168	166	181	178	167	154	150	154	157	169	178	173	163	167	187	135	94	168	253
255	255	232	217	243	225	162	125	81	153	173	173	188	191	173	162	148	139	143	154	166	182	192	182	173	182	115	79	141	242
255	255	233	221	231	183	142	106	71	136	185	174	198	184	168	150	126	119	119	134	156	175	197	203	179	182	110	75	140	245
255	252	208	218	217	163	149	94	71	116	187	188	186	155	148	125	107	103	100	111	134	154	163	199	195	161	100	81	117	225
255	244	189	213	214	171	141	114	76	84	158	206	189	140	136	116	101	99	94	109	120	138	150	197	223	136	98	80	105	222
255	240	181	202	196	145	102	131	83	79	145	210	174	143	133	111	100	84	85	99	115	135	142	181	230	221	109	91	118	238
255	236	178	172	196	183	75	116	101	84	167	200	153	104	92	66	65	71	70	63	74	101	113	174	220	226	102	113	129	234
255	237	180	171	207	197	71	72	111	85	156	186	155	93	82	43	31	28	28	37	67	93	126	179	211	201	97	116	81	188
255	246	197	194	235	175	48	53	105	92	145	180	149	146	90	53	35	22	24	48	69	98	179	175	203	178	101	78	63	182
255	254	215	190	236	172	70	57	77	87	131	180	142	156	108	22	36	36	43	47	19	118	189	162	206	164	65	68	59	187
255	248	178	180	229	177	72	58	61	68	114	182	154	138	154	81	56	56	58	69	96	185	157	163	221	148	52	69	90	219
255	210	132	203	229	175	103	60	56	54	109	169	163	143	137	145	101	88	85	106	165	150	157	195	254	200	68	63	150	255
209	150	123	226	236	173	148	82	58	45	146	217	205	180	142	150	164	146	144	161	155	180	231	253	255	250	136	68	164	255
203	152	144	197	224	174	156	105	71	69	177	249	255	247	209	166	145	133	128	143	179	236	255	255	255	255	244	186	212	255
254	240	212	165	168	170	149	140	131	144	159	180	224	251	255	255	215	152	152	169	241	255	255	255	255	255	255	255	255	255
255	255	255	225	169	160	125	112	112	158	159	156	160	187	229	255	232	102	96	142	230	255	255	255	255	255	255	255	255	255
255	255	255	255	222	167	118	88	86	132	166	157	161	156	161	192	200	103	83	114	172	233	255	255	255	255	255	255	255	255
255	255	255	255	253	190	114	87	86	91	142	183	229	219	189	160	156	113	81	98	122	180	244	255	255	255	255	255	255	255
255	255	255	255	255	217	108	96	80	91	117	215	255	255	253	226	179	116	77	101	103	156	243	255	255	255	255	255	255	255
255	255	255	255	255	249	177	141	112	147	204	247	255	255	255	255	245	200	153	151	167	212	255	255	255	255	255	255	255	255
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	255

Digital Image

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

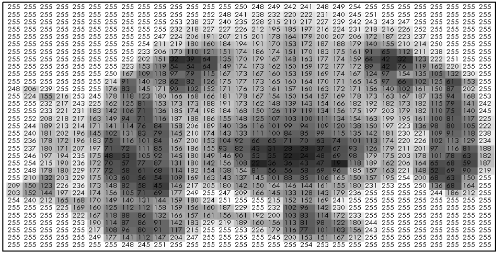


Digital Image

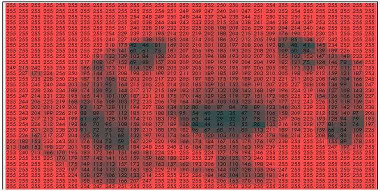
- Image is a matrix with integer values
- We will typically denote it with I
- $I(i,j)$ is called **intensity**
- Matrix I can be $m \times n$ (grayscale)

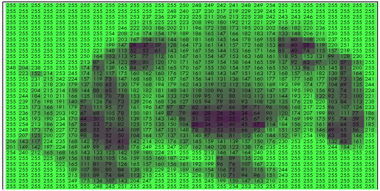
255	255	255	255	255	255	255	255	255	255	255	255	255	255	250	248	249	242	241	248	249	254	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	252	248	241	238	232	220	222	231	240	245	251	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	253	238	237	240	235	228	215	210	217	227	239	242	243	243	247	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	232	218	227	227	226	212	195	185	197	216	224	231	218	216	226	252	255	255	255	255
255	255	255	255	255	255	255	255	255	247	224	206	191	207	215	201	178	164	179	200	207	206	172	187	223	237	255	255	255	255
255	255	255	255	255	255	255	255	254	211	219	180	160	184	194	191	170	153	172	187	188	179	140	155	210	214	250	255	255	255
255	255	255	255	255	255	255	255	233	206	170	110	121	151	174	186	174	151	170	183	175	161	91	65	112	211	238	255	255	255
255	255	255	255	255	255	255	252	202	151	32	39	64	135	170	179	167	148	163	177	174	159	64	42	32	123	222	251	255	255
255	255	255	255	255	255	255	223	153	119	54	54	64	149	174	173	162	150	159	172	177	172	89	42	76	119	162	220	255	255
255	255	255	255	255	255	250	167	109	118	97	79	115	167	173	167	160	153	159	169	174	167	124	97	154	135	105	132	230	255
255	255	255	255	255	255	214	91	140	128	62	82	126	175	177	173	165	160	164	170	171	165	145	97	66	102	125	61	153	255
248	206	239	255	255	255	176	83	145	171	90	102	152	171	176	173	161	157	160	163	172	171	156	140	102	161	150	87	202	255
255	224	155	216	253	245	178	118	123	180	166	168	166	181	178	167	154	150	154	157	169	178	173	163	167	187	135	94	168	253
255	255	232	217	243	225	162	125	81	153	173	173	188	191	173	162	148	139	143	154	166	182	192	182	173	182	115	79	141	242
255	255	233	221	231	183	142	106	71	136	185	174	198	184	168	150	126	119	119	134	156	175	197	203	179	182	110	75	140	245
255	252	208	218	217	163	149	94	71	116	187	188	186	155	148	125	107	103	100	111	134	154	163	199	195	161	100	81	117	225
255	244	189	213	214	171	141	114	76	84	158	206	189	140	136	116	101	99	94	109	120	138	150	197	223	136	98	80	105	222
255	240	181	202	196	145	102	131	83	79	145	210	174	143	133	111	100	84	85	99	115	135	142	181	230	221	109	91	118	238
255	236	178	172	196	183	75	116	101	84	167	200	153	104	92	66	65	71	70	63	74	101	113	174	220	226	102	113	129	234
255	237	180	171	207	197	71	72	111	85	156	186	155	93	82	43	31	28	28	37	67	93	126	179	211	201	97	116	81	188
255	246	197	194	235	175	48	53	105	92	145	180	149	146	90	53	35	22	24	48	69	98	179	175	203	178	101	78	63	182
255	254	215	190	236	172	70	57	77	87	131	180	142	156	108	22	36	36	43	47	19	118	189	162	206	164	65	68	59	187
255	248	178	180	229	177	72	58	61	68	114	182	154	138	154	81	56	56	58	69	96	185	157	163	221	148	52	69	90	219
255	210	132	203	229	175	103	60	56	54	109	169	163	143	137	145	101	88	85	106	165	150	157	195	254	200	68	63	150	255
209	150	123	226	236	173	148	82	58	45	146	217	205	180	142	150	164	146	144	161	155	180	231	253	255	250	136	68	164	255
203	152	144	197	224	174	156	105	71	69	177	249	255	247	209	166	145	133	128	143	179	236	255	255	255	255	244	186	212	255
254	240	212	165	168	170	149	140	131	144	159	180	224	251	255	255	215	152	152	169	241	255	255	255	255	255	255	255	255	255
255	255	255	225	169	160	125	112	112	158	159	156	160	187	229	255	232	102	96	142	230	255	255	255	255	255	255	255	255	255
255	255	255	255	222	167	118	88	86	132	166	157	161	156	161	192	200	103	83	114	172	233	255	255	255	255	255	255	255	255
255	255	255	255	253	190	114	87	86	91	142	183	229	219	189	160	156	113	81	98	122	180	244	255	255	255	255	255	255	255
255	255	255	255	255	217	108	96	80	91	117	215	255	255	253	226	179	116	77	101	103	156	243	255	255	255	255	255	255	255
255	255	255	255	255	249	177	141	112	147	204	247	255	255	255	255	245	200	153	151	167	212	255	255	255	255	255	255	255	255
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	255

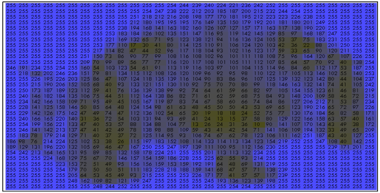
- Image is a matrix with integer values
- We will typically denote it with I
- $I(i,j)$ is called **intensity**
- Matrix I can be $m \times n$ (grayscale)
- or $m \times n \times 3$ (color)



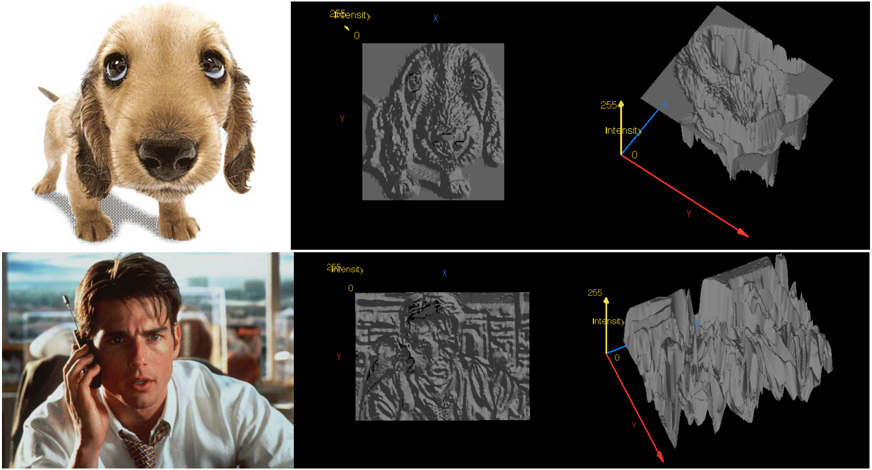
- Image is a matrix with integer values
- We will typically denote it with I
- $I(i,j)$ is called **intensity**
- Matrix I can be $m \times n$ (grayscale)
- or $m \times n \times 3$ (color)







Intensity



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

Image Transformations

- As with any function, we can apply operators to an image, e.g.:

$I(i,j)$ $\quad$ $J(i,j) = I(i,j) + 50$ $\quad$ $J(i,j) = I(i,-j)$ $\quad$ $I(i,j) \cdot (I(i,j) < 250)$

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

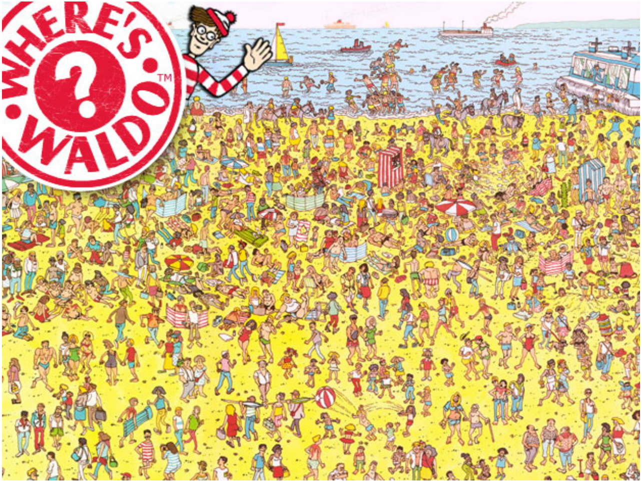
[Adapted from: N. Snavely]

Linear Filters

Reading: Szeliski book, Chapter 3.2

Motivation: Finding Waldo

- How can we find Waldo?



[Source: R. Urtasun]

Answer

- Slide and compare!
- In formal language: **filtering**

Motivation: Noise reduction

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

[Source: S. Seitz]

Image Filtering

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

10	5	3
4	5	1
1	1	7

Local image data

Some function →

	7	

Modified image data

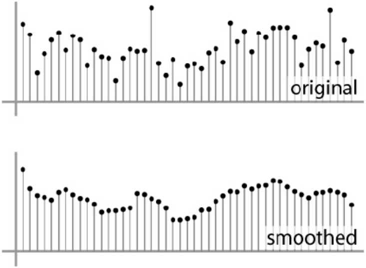
[Source: L. Zhang]

Applications of Filtering

- Detect patterns, e.g., **template matching**.
- Enhance an image, e.g., **denoise, resize**.
- Extract information, e.g., **texture, edges**.

Noise reduction

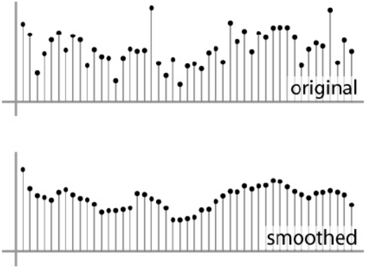
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



[Source: S. Marschner]

Noise reduction

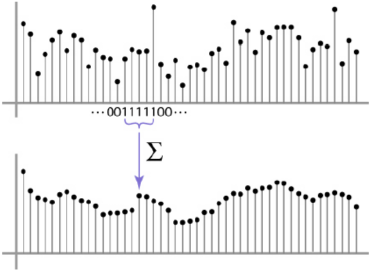
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



[Source: S. Marschner]

Noise reduction

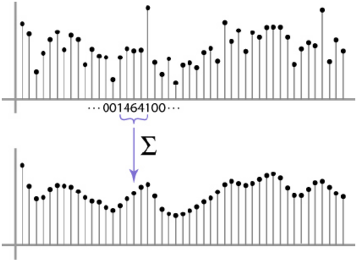
- Simplest thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- **Moving average** in 1D: [1, 1, 1, 1, 1]/5



[Source: S. Marschner]

Noise reduction

- 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



[Source: S. Marschner]

Moving Average in 2D

$I(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(i,j)$

	0								

[Source: S. Seitz]

Moving Average in 2D

$I(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(i,j)$

	0	10							

[Source: S. Seitz]

Moving Average in 2D

$I(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(i,j)$

	0	10	20						

[Source: S. Seitz]

Moving Average in 2D

$I(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(i,j)$

	0	10	20	30					

[Source: S. Seitz]

Moving Average in 2D

$I(i, j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(i, j)$

	0	10	20	30	30				

[Source: S. Seitz]

Moving Average in 2D

$I(i, j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(i, j)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

[Source: S. Seitz]

Linear Filtering: Correlation

- 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

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

Linear Filtering: Correlation

- 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**.

Linear Filtering: Correlation

- 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

$$G = F \otimes I$$

Linear Filtering: Correlation

- 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

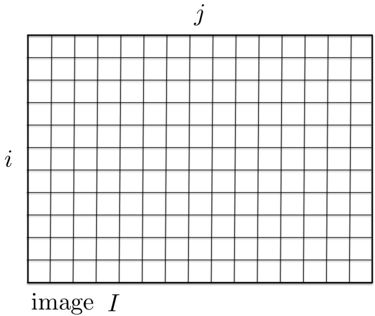
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

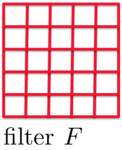
- The entries of the weight **kernel** or **mask** $F(u,v)$ are often called the **filter coefficients**.
- This operator is the **correlation** operator

$$G = F \otimes I$$

Linear Filtering: Correlation

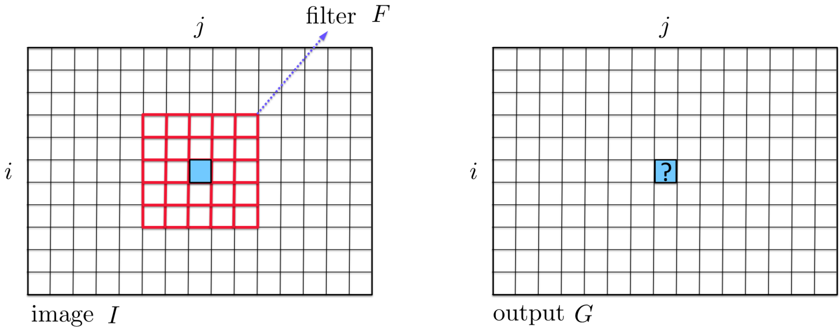
- It's really easy!





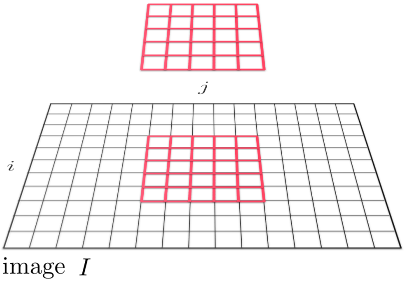
Linear Filtering: Correlation

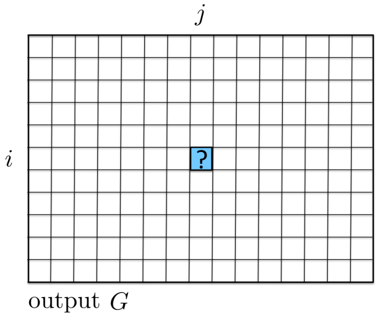
- It's really easy!



Linear Filtering: Correlation

- It's really easy!





Linear Filtering: Correlation

- It's really easy!

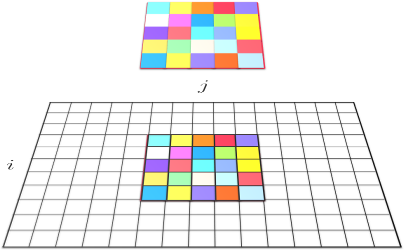
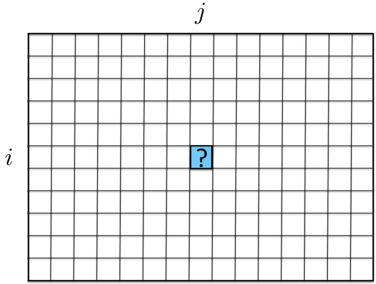


image I



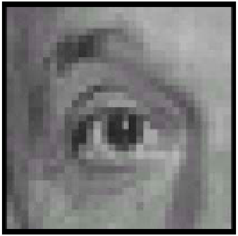
output G

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

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

Example

- What's the result?



Original

0	0	0
0	1	0
0	0	0

?

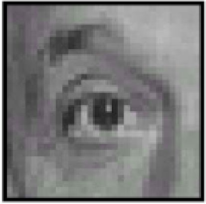
[Source: D. Lowe]

Example

- What's the result?

Original

0	0	0
0	1	0
0	0	0

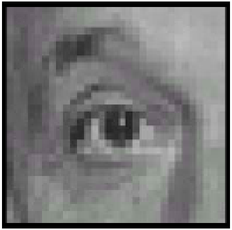


Filtered
(no change)

[Source: D. Lowe]

Example

- What's the result?



Original

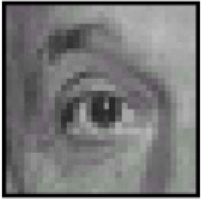
0	0	0
0	0	1
0	0	0

?

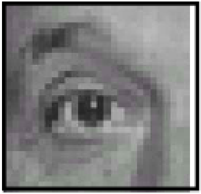
[Source: D. Lowe]

Example

- What's the result?



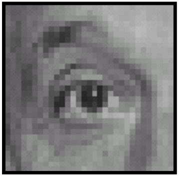
0	0	0
0	0	1
0	0	0



[Source: D. Lowe]

Example

- What's the result?



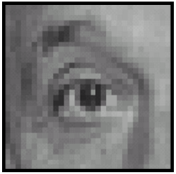
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

[Source: D. Lowe]

Example

- What's the result?



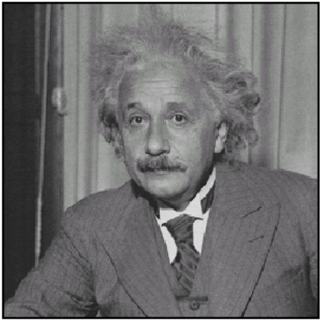
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

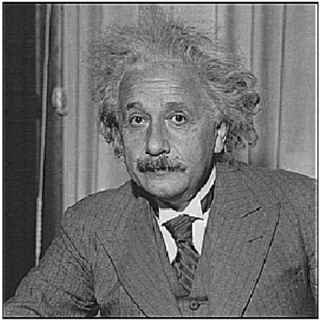
Sharpening filter
(accentuates edges)

[Source: D. Lowe]

Sharpening



before

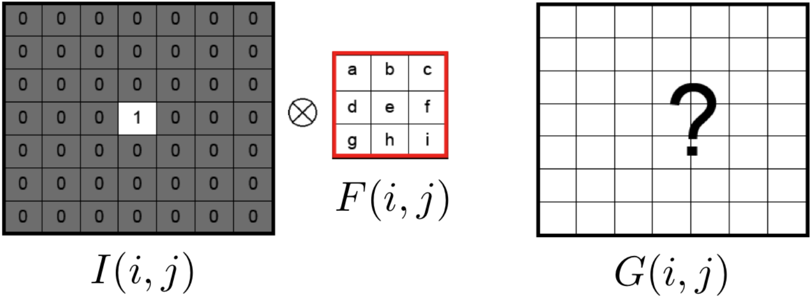


after

[Source: D. Lowe]

Example of Correlation

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

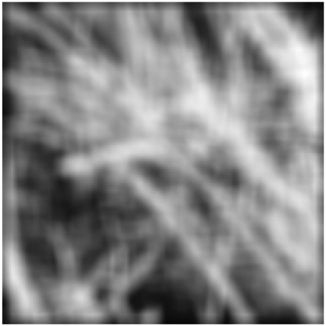


[Source: K. Grauman]

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]

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).

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$I(i,j)$

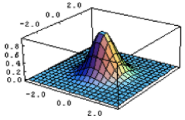
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$F(i,j)$

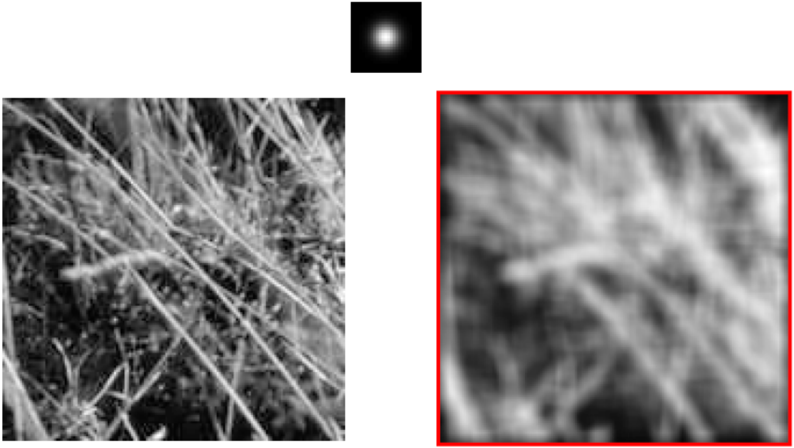
This kernel is an approximation of a 2d Gaussian function:

$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



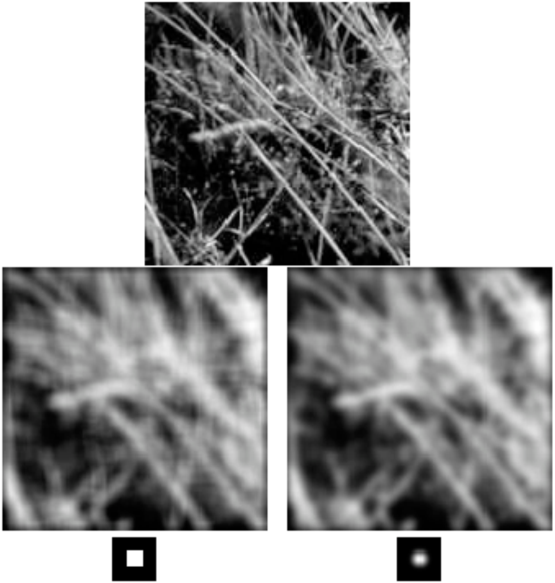
[Source: S. Seitz]

Smoothing with a Gaussian



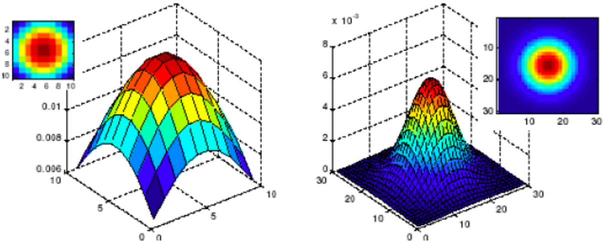
[Source: K. Grauman]

Mean vs Gaussian



Gaussian filter: Parameters

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



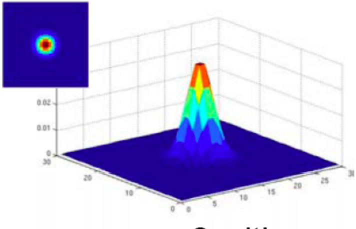
σ = 5 with 10 x 10 kernel

σ = 5 with 30 x 30 kernel

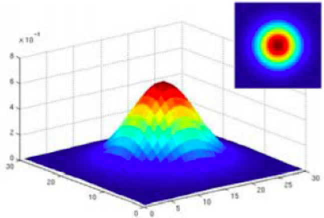
[Source: K. Grauman]

Gaussian filter: Parameters

- **Variance of the Gaussian**: determines extent of smoothing.



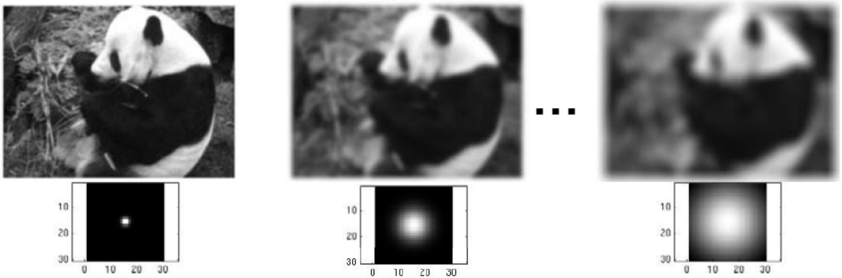
σ = 2 with
30 x 30
kernel



σ = 5 with
30 x 30
kernel

[Source: K. Grauman]

Gaussian filter: Parameters



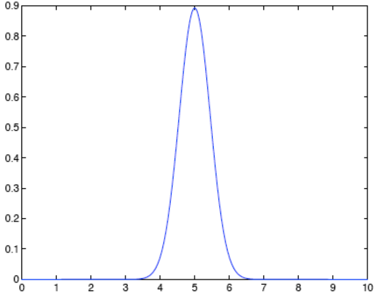

```
for sigma=1:3:10
   h = fspecial('gaussian', fsize, sigma);
   out = imfilter(im, h);
   imshow(out);
   pause;
end
```

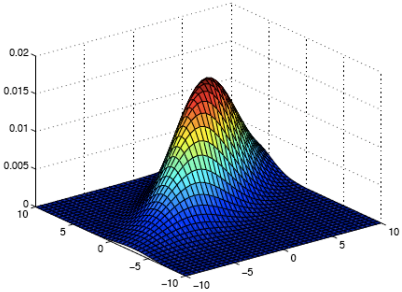
[Source: K. Grauman]

Is this the most general Gaussian?

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

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





- But the simplified version is typically use for filtering.

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.

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.

Finding Waldo

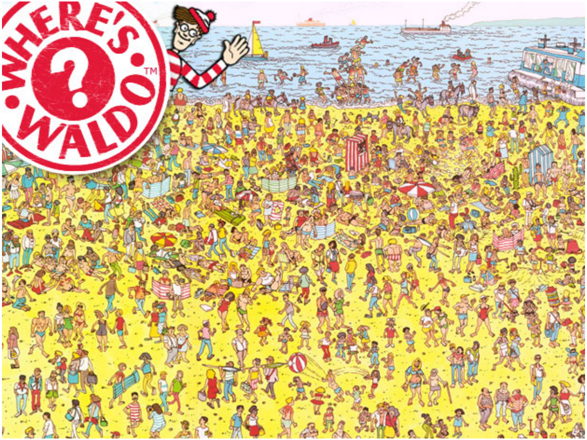
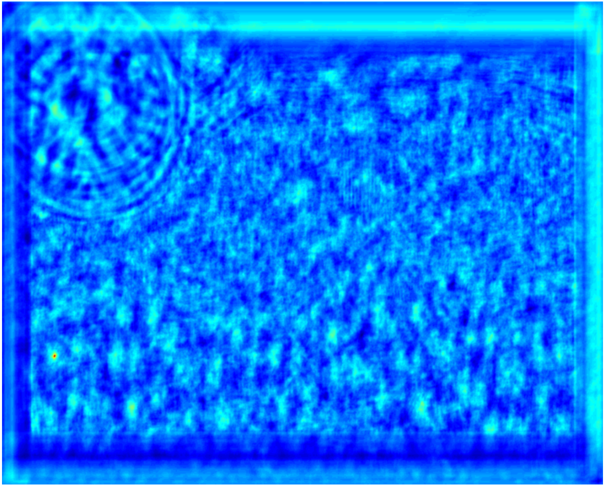


image I

filter F

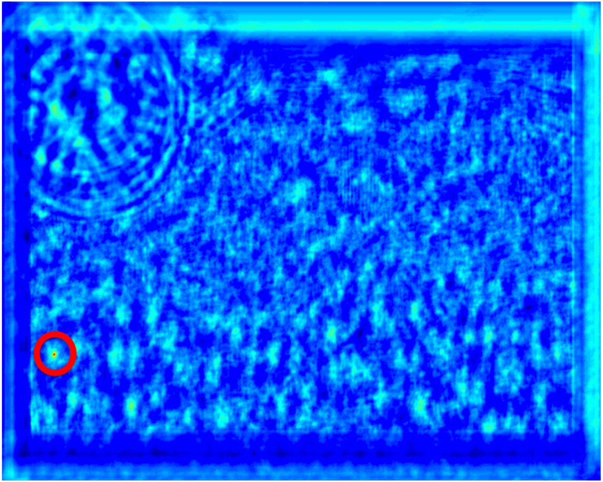
- Let's do normalized cross-correlation

Finding Waldo



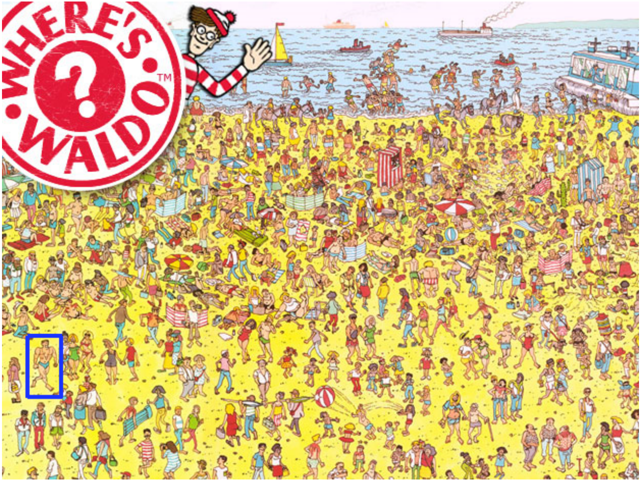
- Result

Finding Waldo



- Result

Finding Waldo



Voila!

- **Homework**: Do it yourself! Code on class webpage. Don't cheat ;)

Convolution

- **Convolution** operator

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

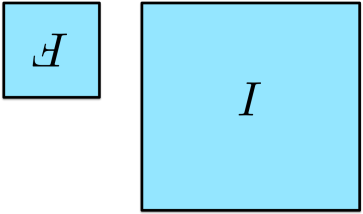
- **Equivalent** to flipping the filter in both dimensions (bottom to top, right to left) and apply correlation.

Convolution

- **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.
- If filter is **symmetric**, then correlation and convolution are **the same**

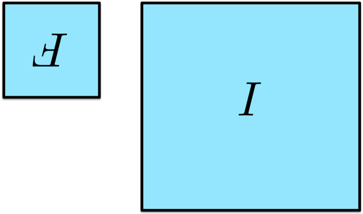


Convolution

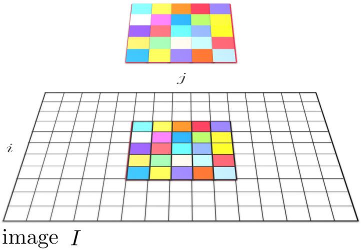
- **Convolution** operator

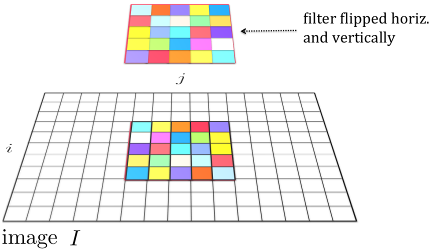
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i-u, j-v)$$

- **Equivalent** to flipping the filter in both dimensions (bottom to top, right to left) and apply correlation.
- If filter is **symmetric**, then correlation and convolution are **the same**



Correlation vs Convolution





- For a Gaussian or box filter, how will the outputs differ?
- 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]

Properties of Convolution

$$\begin{aligned} \text{Commutative} &: f * g = g * f \\ \text{Associative} &: f * (g * h) = (f * g) * h \\ \text{Distributive} &: f * (g + h) = f * g + f * h \\ \text{Assoc. with scalar multiplier} &: \lambda \cdot (f * g) = (\lambda \cdot f) * h \end{aligned}$$

- 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

$$\begin{aligned} \text{Commutative} &: f * g = g * f \\ \text{Associative} &: f * (g * h) = (f * g) * h \\ \text{Distributive} &: f * (g + h) = f * g + f * h \\ \text{Assoc. with scalar multiplier} &: \lambda \cdot (f * g) = (\lambda \cdot f) * h \end{aligned}$$

- 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.

Properties of Convolution

$$\begin{aligned} \text{Commutative} &: f * g = g * f \\ \text{Associative} &: f * (g * h) = (f * g) * h \\ \text{Distributive} &: f * (g + h) = f * g + f * h \\ \text{Assoc. with scalar multiplier} &: \lambda \cdot (f * g) = (\lambda \cdot f) * h \end{aligned}$$

- 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.

Gaussian Filter

- Convolution with itself is another Gaussian

- 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]

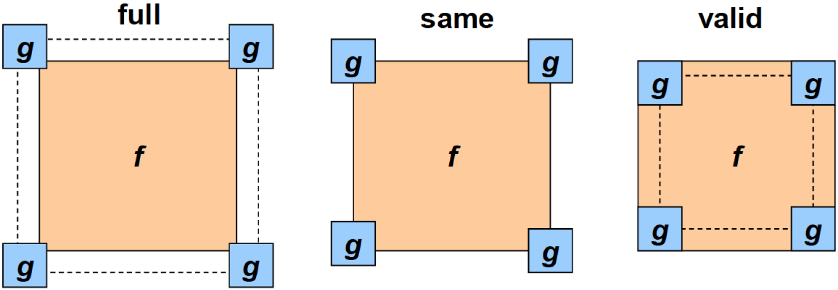
Boundary Effects

- 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]

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]

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)?

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

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

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

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

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

Separable Filters: Speed-up Trick!

- 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?

Separable Filters: Speed-up Trick!

- 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** $2K$ **operations**.

Separable Filters: Speed-up Trick!

- 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** $2K$ **operations**.
- If this is possible, then the convolution filter is called **separable**.

Separable Filters: Speed-up Trick!

- 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** $2K$ **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]

Separable Filters: Speed-up Trick!

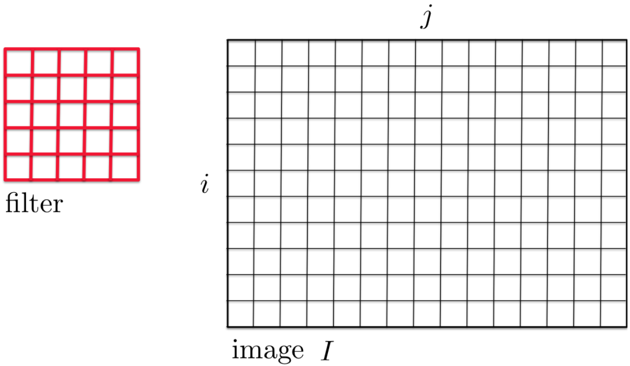
- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?
- In many cases (**not all!**), this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution,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

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

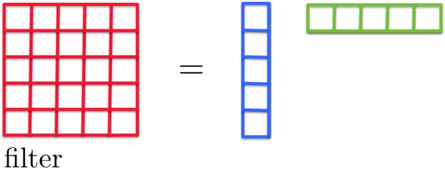
- **Homework:** Think **why** in the case of separable filters 2D convolution is the same as two 1D convolutions

[Source: R. Urtasun]

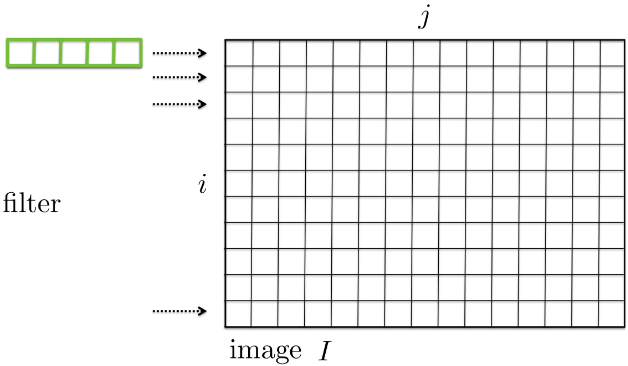
How it Works



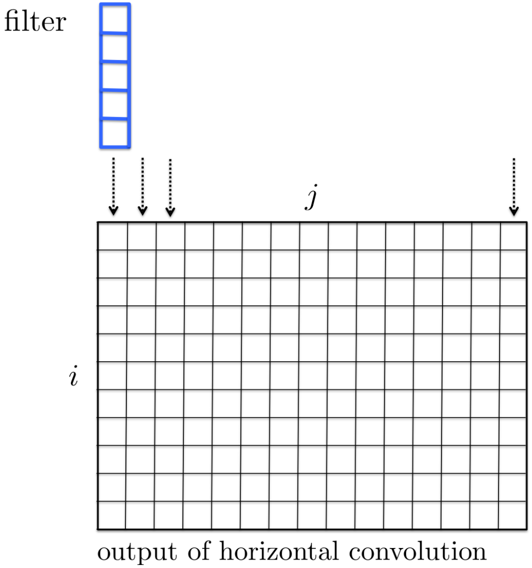
How it Works



How it Works



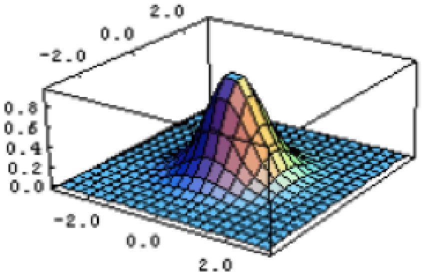
How it Works



Separable Filters: Gaussian filters

- One famous separable filter we already know:

$$\text{Gaussian} \; : \; f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}$$
$$= \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{\sigma^2}}\right) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{\sigma^2}}\right)$$



Let's play a game...

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

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

[Source: R. Urtasun]

Let's play a game...

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

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \qquad \frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

What does this filter do?

[Source: R. Urtasun]

Let's play a game...

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

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$$

[Source: R. Urtasun]

Let's play a game...

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

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

What does this filter do?

[Source: R. Urtasun]

Let's play a game...

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

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$$

[Source: R. Urtasun]

Let's play a game...

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

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array}$$

What does this filter do?

[Source: R. Urtasun]

How can we tell if a given filter F is indeed separable?

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

How can we tell if a given filter F is indeed separable?

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

How can we tell if a given filter F is indeed separable?

- 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 $\Sigma = \mathrm{diag}(\sigma_i)$.

- Matlab: $[U,S,V] = \textsc{svd}(F);$

How can we tell if a given filter F is indeed separable?

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

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

[Source: R. Urtasun]

How can we tell if a given filter F is indeed separable?

- 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 $\Sigma = \text{diag}(\sigma_i)$.
- Matlab: $[\mathrm{U,S,V}] = \textsc{svd}(\mathrm{F});$
- $\sqrt{\sigma_1}\mathbf{u}_1$ and $\sqrt{\sigma_1}\mathbf{v}_1^T$ are the vertical and horizontal filter.

[Source: R. Urtasun]

Sharpening revisited

- What does blurring take away?

−

=

- Let's add it back

+ α

=

[Source: S. Lazebnik]

Sharpening

[Source: N. Snavely]

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
- CORR2, FILTER2: correlation, NORMXCORR2 normalized correlation
- CONV2: does correlation
- FSPECIAL: creates special filters including a Gaussian

Next time:

Edge Detection