

Welcome to
Computer Vision
Week #2 - Lecture

Agenda

Week #2 - Computer Vision - Lecture Image Processing for Computer Vision

1

Linear Image processing: Images as function

2

Linear Image processing: Filtering

Agenda

Week #2 - Computer Vision - Lecture Image Processing for Computer Vision

1

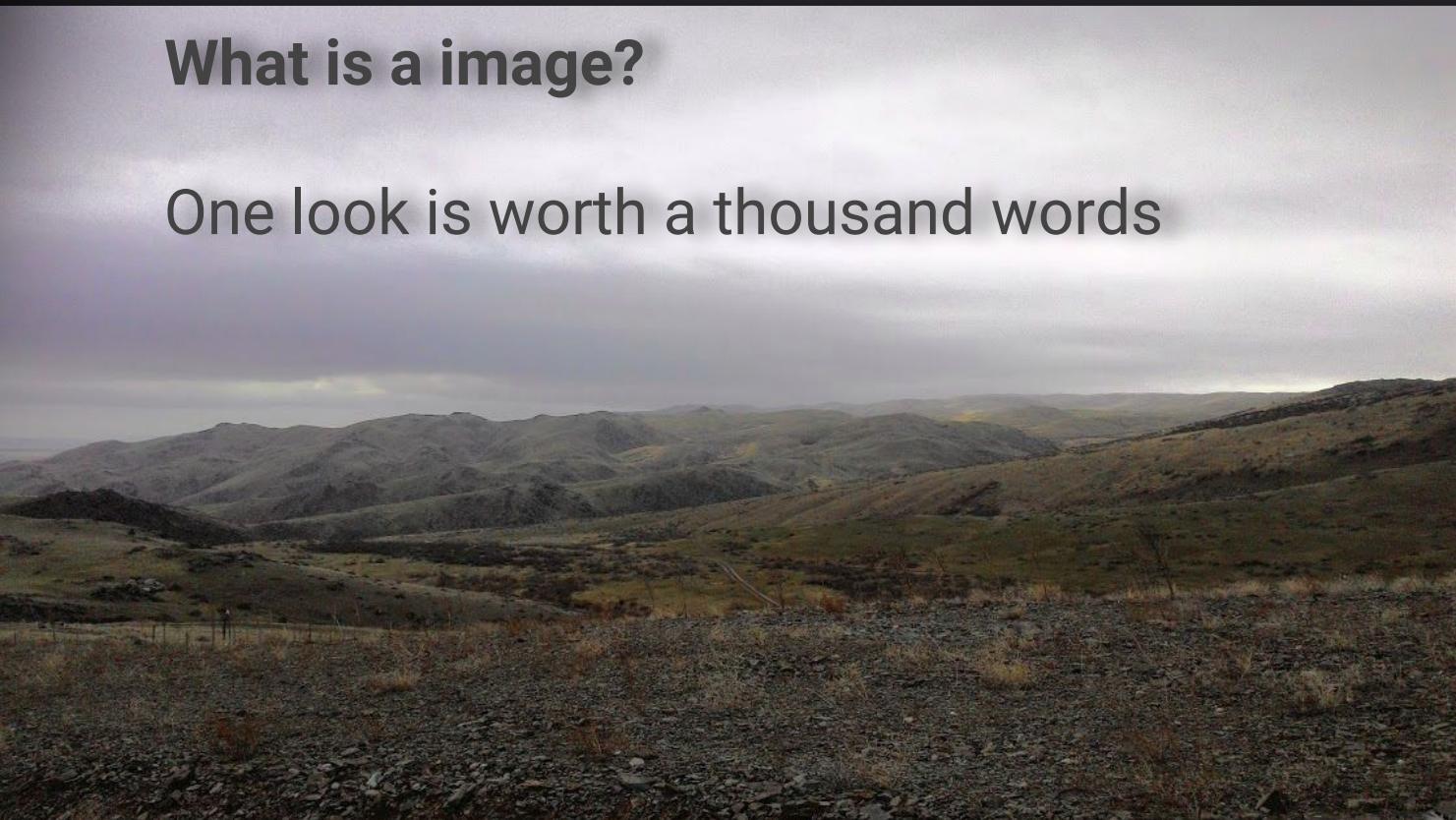
Linear Image processing: Images as function

2

Linear Image processing: Filtering

What is a image?

One look is worth a thousand words



What is a image?

A (color) image is a 3D tensor of numbers



Combination of 3 channels

Red

Green

Blue



Grayscale Image

An Image contains discrete number of pixels

Pixel value: grayscale range [0, ..., 255]

0 is pure black while 255 is pure white



Grayscale Image

Pixel values samples



x=831.571 y=206.583 [237]



Color Image

Each pixel contains 3 numbers [R, G, B]
ranging [0, ..., 255]

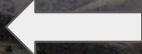


Color Image

Pixel values samples



x=777.309 y=191.079 [250, 248, 249]



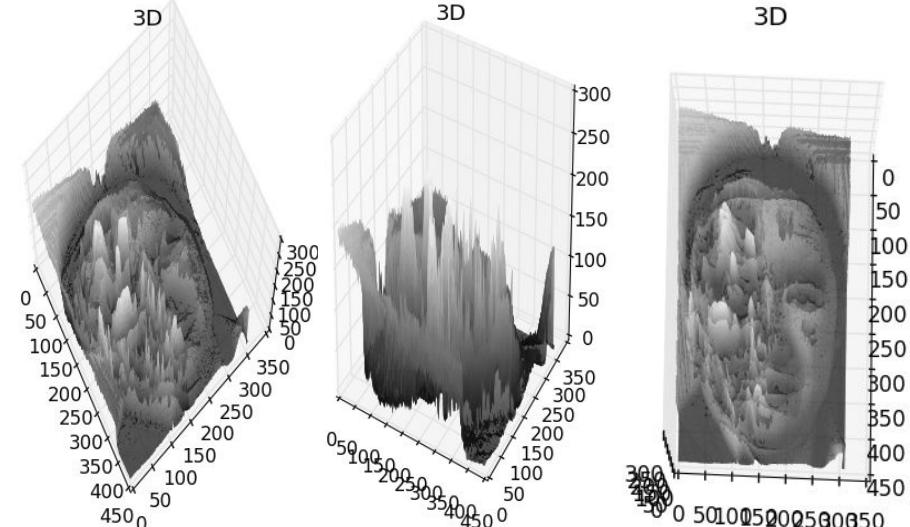
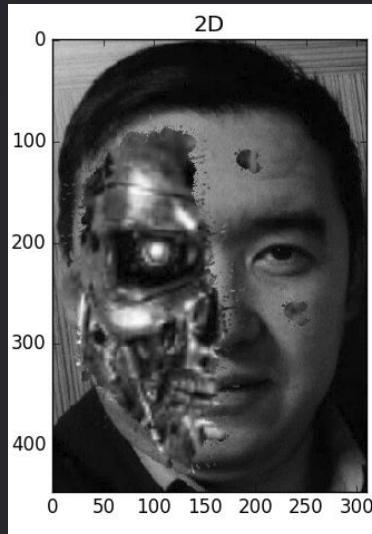
x=95.154 y=387.088 [11, 9, 10]



x=670.999 y=515.546 [93, 89, 86]

An Image as function f from \mathbb{R}^2 to \mathbb{R}^M :

$f(x, y)$ gives the intensity at position (x, y)



An Image as function f from \mathbb{R}^2 to \mathbb{R}^M :

$f(x, y)$ gives the intensity at position (x, y)

Grayscale Image has single channel

Color Image has three channels

An Image as function f from \mathbb{R}^2 to \mathbb{R}^M :

$f(x, y)$ gives the intensity at position (x, y)

Grayscale Image can be represented as 2D Matrix with one channel

Color Image can be represented as 2D Matrix with three channels

An Image as function f from \mathbb{R}^2 to \mathbb{R}^M :

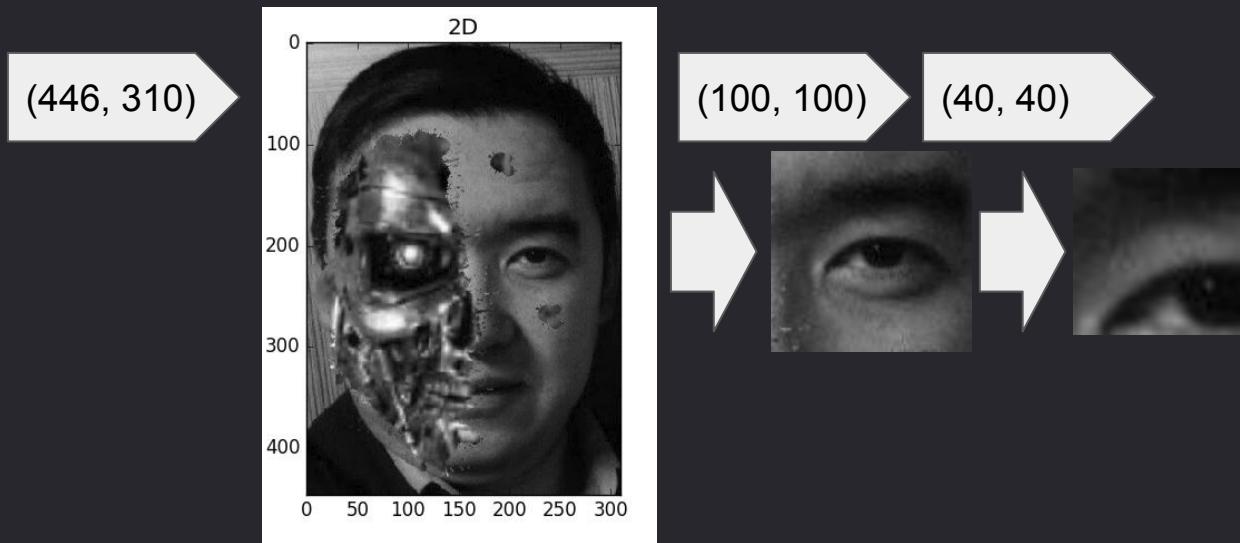
$f(x, y)$ gives the intensity at position (x, y)

Grayscale Image can be represented as 2D Matrix with one channel

Color Image can be represented as 2D Matrix with three channels

An Image as function f from \mathbb{R}^2 to \mathbb{R}^M :

$f(x, y)$ gives the intensity at position (x, y)



An Image as function f from \mathbb{R}^2 to \mathbb{R}^M :

$f(x, y)$ gives the intensity at position (x, y)

(40, 40)



How we see
grayscale image

```
@jomart-hp-pc: ~/Desktop
jonart@jomart-hp-pc:~/Desktop$ python crop.py
[[ 50  51  50  45  45  38  31  33  33  26  21
 19  20  17  12  4  5  11  11  12  15  14
 4  4  2  1]
[ 55  49  47  45  43  41  37  38  36  20  12
 13  16  16  16  16  17  18  12
 16  17  18  18  16  16  11  8  7  8  9
 7  4  3  4  3  2  3]
[ 48  47  49  47  43  41  36  36  33  24  17
 16  13  7  11  16  19  17  11  6  7  10
 16  13  5  5  5  7]
[ 41  43  44  46  46  35  31  32  30  24  18
 14  12  5  8  14  12  14  11  8  8  8
 14  12  5  4  5  7]
[ 41  43  44  38  38  32  32  29  29  26  21
 21  18  17  17  17  16  14  15  7
 11  18  11  7  11  13  9  11  12  10  5
 3  6  9  5  5  6  6]
[ 43  41  39  33  29  29  26  26  23  24  23
 20  19  20  19  17  21  13
 11  16  13  10  16  18  15  14  12  10  8
 7  8  9  8  8  8  10
 11  11  11  11]
[ 40  39  39  36  32  28  23  23  23  26  26
 23  21  21  21  21  21  20  17
 12  15  11  18  17  18  15  12  11  12
 13  12  9  11  11  11  13]
[ 36  37  49  38  32  28  25  27  28  26  24
 24  24  24  24  23  23  24  21
 14  15  15  12  18  19  15  17  18  18
 18  19  28  26  16  16
 20  21  23  25]
[ 35  36  36  36  36  36  36  36  36  36  36
 36  36  36  36  36  36  36  36
 22  27  22  18  22  26  22  18  24
 19  19  22  26  25  21  19  22
 24  25  28  28  30]
[ 38  33  29  24  19  22  29  21  31  28
 28  21  24  25  25  28  29  22
 25  30  26  23  26  29  25  21  26
 26  22  28  26  23  27  28
 28  31  27  23  25]
[ 34  27  23  23  24  27  30  39  27  27
 20  25  28  29  28  29  28  29
 33  27
 37  37  37  38]
[ 25  23  26  31  33  31  29  29  27  27
 30  28  31  36  31  36  31  33
 34  34  34  34  34  35  37  40
 49  49  50]
[ 25  26  29  32  31  27  28  33  33  36
 28  28  31  28  34  40  30  32
 36  38  38  40  44  47  49  48  45
 46  42  42  43  45  45  52  57
 58  59  62]
[ 31  29  27  28  31  30  31  36  36  32
 30  32  35  30  36  43  32  38
 44  48  58  53  54  54  59  62  58
 61  60  58  56  56  53  51  56  69]
```

First line - grayscale values

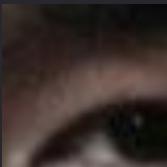
Second line - grayscale values

How computer see

An Image as function f from \mathbb{R}^2 to \mathbb{R}^M :

$f(x, y)$ gives the 3 values at position (x, y)

(40, 40)



How we see
color image

```
jomart@jomart-hp-pc: ~/Desktop
[jonart@jomart-hp-pc:~/Desktop]$ python crop.py
[[ 64  53  49]
 [ 59  48  44]
 [ 57  48  43]
 [ 55  46  41]
 [ 52  42  40]
 [ 45  35  33]
 [ 38  30  27]
 [ 39  31  28]
 [ 37  32  29]
 [ 30  25  22]
 [ 24  20  17]
 [ 23  19  16]
 [ 21  20  18]
 [ 20  19  17]
 [ 20  19  17]
 [ 20  19  17]
 [ 16  17  12]
 [ 14  15  10]
 [ 19  19  17]
 [ 20  20  18]
 [ 17  17  15]
 [ 12  12  10]
 [  4   4   2]]
```

First line - First pixel values

First line - Second pixel values

How computer see

An Image as function f from \mathbb{R}^2 to \mathbb{R}^M :

We can convert values in each location



Image as function |

An Image as function f from R^2 to R^M :

We can convert values in each location



```
:~$ python crop.py
[ 5 196 197 199 200 194 197 196 194 197 282 201 195 191 1
 8 197 192 193 187 189 183 197 192 191 178 177 188 178 1
 84 178 180 179 176 175 177 176 173 172 173 174 175 174 1
 176 179 178 174 169 163 165 178 163 164 162 163 164 164 1
 166 163 161 156 162 160 166 156 151 153 157 152 143 147 1
 8 151 147 142 138 138 141 144 144 143 133 135 141 138 142 148 1
 48 142 137 136 148 148 147 148 135 137 129 128 128 128 128 1
 122 124 118 124 116 114 113 121 118 117 116 169 168 98 76 1
 50 123 122 121 120 119 118 117 116 115 114 113 112 111 109 1
 43 29 23 28 40 48 48 38 36 49 55 49 37 32 59 76
 1 92 108 111 111 111 105 121 118 120 120 125 134 126 122 134 1
 44 130 128 126 129 133 133 130 127 135 115 126 125 121 134 130 1
 130 121 116 129 133 122 117 119 121 120 117 117 119 119 117 119 1
 128 118 117 120 113 114 109 113 118 114 167 110 174 164 166 166 1
 6 103 102 105 97 116 108 102 105 101 98 185 166 183 184 99 102 1
 35 104 97 96 97 106 102 102 99 96 92 95 95 93 93 95 95 95
 94 95 96 96 93 91 93 95 93 96 87 84 95 98 85 101 101
 92 99 103 101 106 196 197 199 195 193 192 194 189 191 193 194 1
 191 187 190 191 190 198 189 191 191 188 184 184 181 178 194 187 187
 173 178 174 176 188 174 178 171 179 176 174 173 174 174 174 173 1
 73 174 179 178 175 174 168 167 168 168 167 168 167 168 165 164 1
 156 155 154 155 155 155 155 155 155 155 155 155 155 155 155 155 1
 151 145 146 144 143 145 147 147 147 147 158 146 147 158 158 146 146 1
 145 142 147 145 149 148 146 139 147 143 143 148 149 144 142 145 133
 127 123 122 126 132 138 133 133 124 127 113 114 111 119 106 107 105
 94 83 73 66 53 44 37 32 49 46 43 48 60 77 92 99 76
 68 73 67 53 39 34 32 28 23 22 27 57 50 27 32 44 48
 37 37 43 57 69 72 74 77 95 92 107 104 100 98 115 113 1
 127 125 134 125 130 130 131 130 128 128 129 130 128 137 125 129 1
 127 133 130 124 122 121 116 116 125 128 124 122 119 121 124 117 117 1
 118 116 119 113 112 12 109 111 114 113 167 163 167 112 116 108 105 1
 100 101 103 104 106 96 95 95 98 103 108 104 103 99 96 91 92
 99 104 105 106 92 85 95 97 99 98 96 98 100 98 103 101 95
 91 98 101 104 101 106 102 101 98 97 99 95 100 98 96 98 107
 94 95 100 101 102 103 104 105 106 107 108 109 108 109 107 106 1
 187 189 187 184 182 183 185 183 180 182 185 185 185 185 1
 183 184 186 177 181 189 181 188 182 178 180 177 176 178 171 1
 171 169 172 173 169 166 164 163 163 164 163 162 162 162 160 1
 155 163 165 159 156 166 161 157 166 161 163 163 159 156 159 163 159
 161 157 153 153 153 156 149 144 143 147 148 146 149 155 156 149 146 1
 153 151 149 150 147 149 141 149 152 153 152 147 153 147 144 148 149 1
 142 142 135 131 127 127 129 128 122 116 117 114 104 103 90 87 82
```

```
:~$ cd Desktop
jonart@jonart-hp-pc:~/Desktop$ python crop.py
[ 5 59 60 59 58 56 55 61 58 59 61 58 53 54 60 64 61
 62 64 62 57 58 63 62 68 66 72 58 63 64 77 78 67 77 74
 69 73 67 71 77 75 76 79 80 78 79 82 83 81 80 81 82
 84 86 82 79 77 81 86 89 96 84 87 91 93 92 91 91 93
 94 99 100 95 92 94 99 103 95 95 99 104 102 98 103 112 108 110
 112 111 107 104 108 113 117 117 114 111 115 122 126 114 117 113 107 112
 121 122 115 113 118 119 115 107 108 115 115 126 118 124 127 127 127 128
 131 132 133 134 135 136 137 135 135 135 130 121 129 133 121 123
 134 135 136 137 138 139 140 135 135 135 130 121 129 133 121 123
 194 199 205 207 204 205 204 197 206 188 180 185 195 200 200 200 200 200
 215 209 212 226 232 227 215 207 217 219 206 200 206 218 223 196 179 178
 177 174 163 147 144 144 144 150 134 137 135 135 130 121 129 133 121 123
 126 111 126 128 128 125 122 121 125 129 120 140 129 138 134 121 125 130
 130 125 134 139 126 122 133 138 136 134 135 138 136 136 136 131 132 139
 130 127 137 138 135 142 141 146 142 147 141 148 145 141 151 151 141 147
 149 152 153 158 158 145 147 153 158 149 147 154 155 147 152 155 156 153 151
 150 151 158 159 158 155 153 153 156 159 164 161 165 162 166 163 163
 161 160 159 159 162 164 162 169 162 159 168 169 159 168 159 170 154 154 161
 163 156 152 154 154 59 59 58 59 60 62 63 64 66 64 66 64 62 62 61 61
 64 68 65 64 65 66 66 64 64 67 67 71 69 77 65 68 68 69 79
 82 72 81 79 78 73 73 76 78 76 77 82 81 80 81 80 82 83
 82 83 84 85 86 87 88 89 89 90 91 92 92 92 92 92 92
 99 99 99 100 100 99 101 101 100 106 108 128 103 101 104 100 108 104 105 105
 104 110 109 109 111 112 118 108 108 108 108 105 109 108 105 109 108 106 106
 110 113 110 106 107 107 109 116 108 112 112 107 106 111 113 110 122 124
 128 132 133 129 123 117 122 122 131 128 142 141 144 136 147 148 159 154
 161 172 182 189 202 211 218 213 206 209 212 207 195 178 163 165 179 188
 187 182 188 202 216 221 223 227 232 233 228 198 205 228 223 211 207 201
 218 218 212 198 186 183 181 178 166 163 148 151 155 157 140 142 142 126
 128 130 119 121 130 125 125 124 125 128 128 126 125 126 118 130 126 126
 128 122 125 131 133 134 139 139 130 127 131 133 133 135 135 138 138 137
 137 139 136 142 143 143 146 144 141 142 148 145 148 143 145 147 154
 155 154 152 151 155 159 160 157 152 147 149 145 155 157 160 165 164 163 168
 156 151 159 155 163 170 168 157 159 159 157 159 155 170 152 154 166 165
 164 157 153 150 150 150 151 153 153 155 155 157 158 156 158 157 159 155 157 151
 164 153 152 150 150 150 151 152 152 152 152 152 152 152 152 152 152 152 152 152
 66 66 66 71 73 72 76 72 72 75 75 75 75 75 75 75 75 75 75 75 75
 72 71 69 78 74 66 74 74 73 76 73 77 75 77 78 77 78 77 78 77 78 77
 84 87 83 84 86 89 90 91 92 91 92 93 93 93 93 93 93 93 93 93 93
 100 92 90 96 99 95 94 98 95 94 92 96 96 99 96 98 92 105 98
 94 98 102 102 102 105 108 116 106 111 112 108 107 109 108 106 108 106 106
 102 104 106 105 108 116 114 106 106 103 102 108 108 111 107 106 110
```

An Image as function f from \mathbb{R}^2 to \mathbb{R}^M :

Image thus represented as matrix of integer values



```
jjomart-hp-pc: ~/Desktop  
jonart@jonart-hp-pc: ~/Desktop$ python crop.py  
197 196 195 195 196 197 199 200 194 197 196 194 197 202 201 195 191 194  
193 191 193 198 197 192 193 187 189 183 197 192 191 197 198 177 188 178 181  
186 181 188 184 178 186 179 176 175 177 176 173 172 173 174 175 174 173  
171 169 173 176 179 178 174 169 166 165 171 168 164 162 163 164 164 162  
161 156 155 160 163 161 156 152 166 166 156 151 153 157 152 143 147 145  
143 144 145 151 147 142 138 138 141 144 146 133 135 147 138 142 147 143  
132 133 149 142 137 136 148 148 147 146 135 137 129 128 128 128 127  
124 122 126 124 126 124 118 114 113 125 113 112 111 109 108 102 98 76 67  
67 66 65 65 65 65 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51  
40 46 43 29 23 28 40 48 38 36 49 55 49 37 32 39 76 77  
78 81 92 100 111 111 111 105 121 118 110 120 120 125 134 126 122 134 132  
129 144 130 128 126 129 133 133 130 127 135 115 126 125 121 134 130 125  
125 130 121 116 129 133 122 117 119 121 129 117 117 117 119 119 117 119 116  
125 128 118 117 129 113 114 109 113 118 114 107 110 114 104 104 106 168  
106 103 102 105 97 108 108 102 105 101 98 105 106 103 100 99 102 184  
105 104 97 96 97 108 102 102 99 96 92 95 95 93 93 95 95 92  
94 95 96 96 93 91 93 95 93 96 87 86 96 92 85 101 101 94  
92 99 103 101 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100  
191 187 190 191 190 189 189 191 191 188 188 184 184 187 178 190 187 187 176  
173 183 174 176 188 177 182 178 177 179 178 174 173 172 171 170 169 168 167 166  
173 174 175 175 175 174 168 178 173 173 168 167 166 165 164 163 162 161 160 159  
156 157 157 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155  
151 145 146 144 143 143 145 147 147 147 147 147 147 147 147 147 147 148 146 146 149  
145 142 145 149 148 148 146 139 147 147 143 143 143 143 143 143 144 142 145 133 131  
127 123 122 126 132 138 133 133 124 127 113 114 111 119 108 107 105 101  
94 83 73 66 53 44 37 42 49 46 43 48 60 77 92 90 76 66  
68 73 67 53 39 34 32 28 23 22 27 57 58 27 32 44 46 54  
37 37 43 57 69 72 74 77 95 92 107 104 100 98 115 113 113 129  
127 125 136 134 125 130 130 131 130 128 128 129 130 128 137 125 129 129  
127 133 130 124 122 121 116 116 125 128 124 122 119 121 129 117 117 118  
118 116 119 113 112 112 109 111 114 113 107 103 107 112 118 108 105 101  
106 101 103 104 106 96 95 98 103 108 106 100 99 95 90 91 92 95  
99 104 105 100 92 85 95 97 99 98 96 96 98 100 103 101 95 90  
91 98 101 101 104 101 100 102 101 98 97 99 95 98 100 98 97 98  
91 98 101 101 104 101 100 102 101 98 97 99 95 98 100 98 97 98  
187 189 187 184 182 183 183 183 184 183 108 108 108 108 108 108 108 108 108 108  
183 184 186 177 181 188 181 188 182 178 188 177 176 178 177 173 171 172  
171 168 172 171 169 166 165 164 163 163 164 163 162 162 162 162 162 162 162 162 162  
155 163 165 159 156 156 166 161 157 166 161 163 163 159 156 159 163 158 157  
161 157 153 153 153 158 149 144 143 143 147 148 146 149 145 155 156 149 146 150  
153 151 149 150 147 139 141 149 152 153 152 147 143 147 144 148 149 145  
142 142 135 131 127 127 129 128 122 116 117 114 104 104 103 99 87 82 86
```

Noise in images:

Noise is just another function that is combined with the original to get a new function

$$I' \rightarrow (x,y) = I \rightarrow (x, y) + \rightarrow (x, y)$$

New Image

Image

Noise

Some types of Noises:

Salt & Pepper noise: random occurrence of black and white pixels



Some types of Noises:

Impulse noise: random occurrence of white pixels



Some types of Noises:

Natural (snow), not Impulse noise



Some types of Noises:

Gaussian noise: random variations in intensity drawn from a Gaussian normal distribution



Agenda

Week #2 - Computer Vision - Lecture Image Processing for Computer Vision

1

Linear Image processing: Images as function

2

Linear Image processing: Filtering

Form a new image whose pixels are a combination original pixel values

Goal: modify or enhance image properties

2D discrete-space systems (filters)



Sample

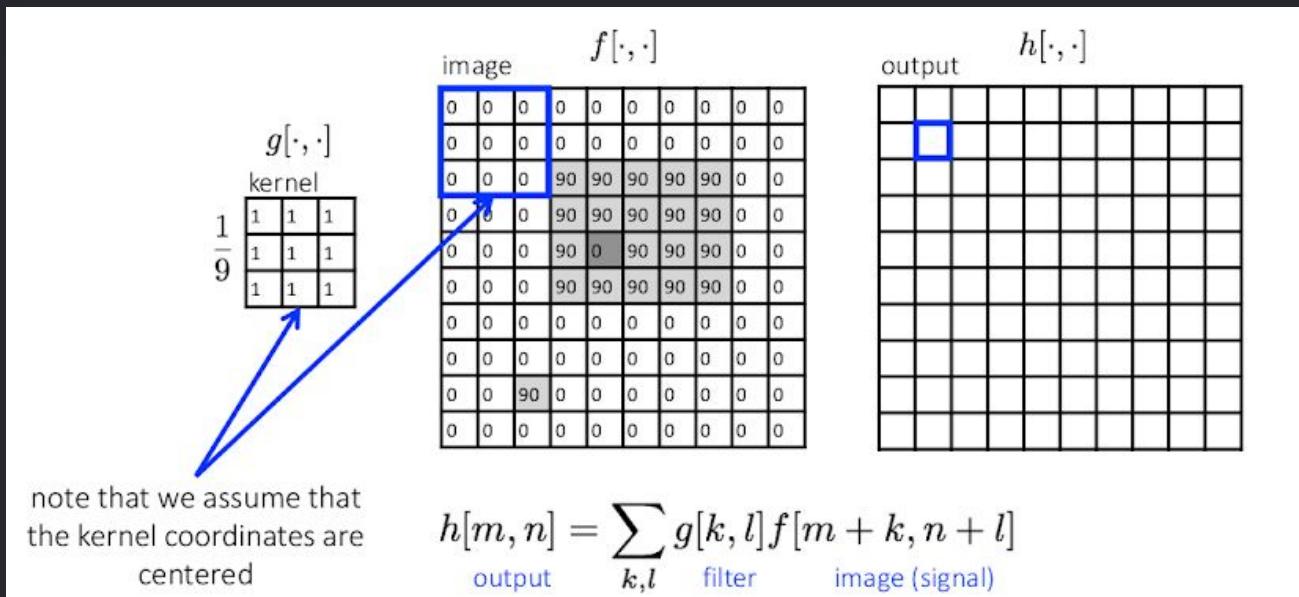
$$f[n, m] \rightarrow \boxed{\text{System } \mathcal{S}} \rightarrow g[n, m]$$

$$g = \mathcal{S}[f], \quad g[n, m] = \mathcal{S}\{f[n, m]\}$$

$$f[n, m] \xrightarrow{\mathcal{S}} g[n, m]$$

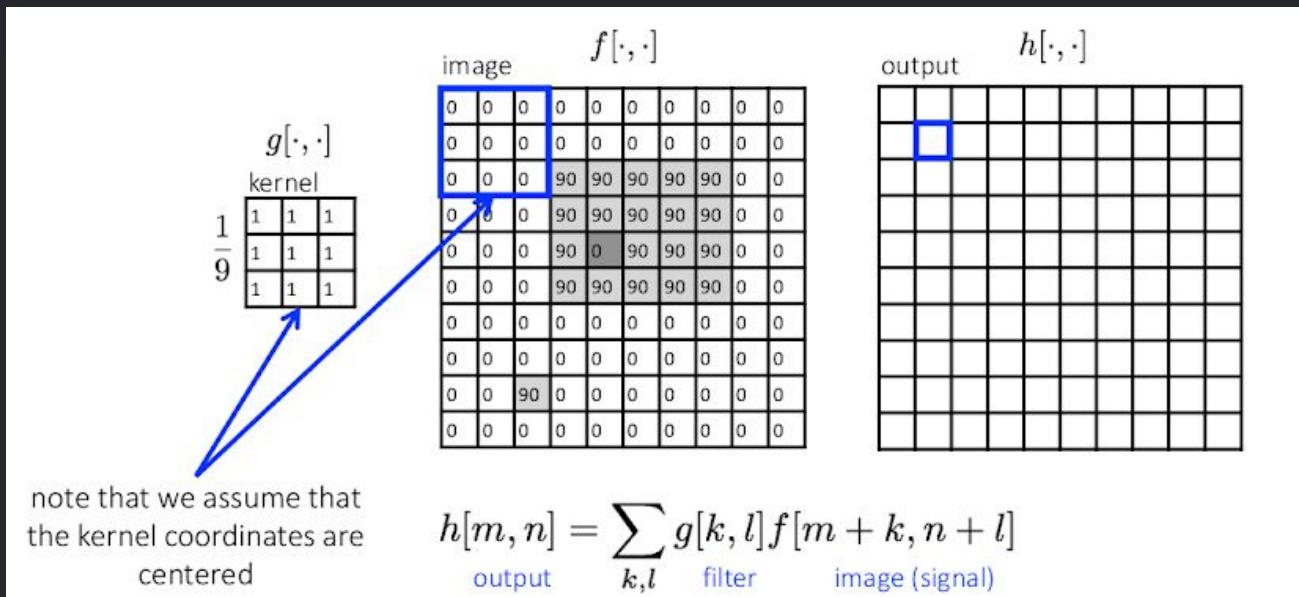
Filtering |

Sample: Box filter - Moving Average in 2D

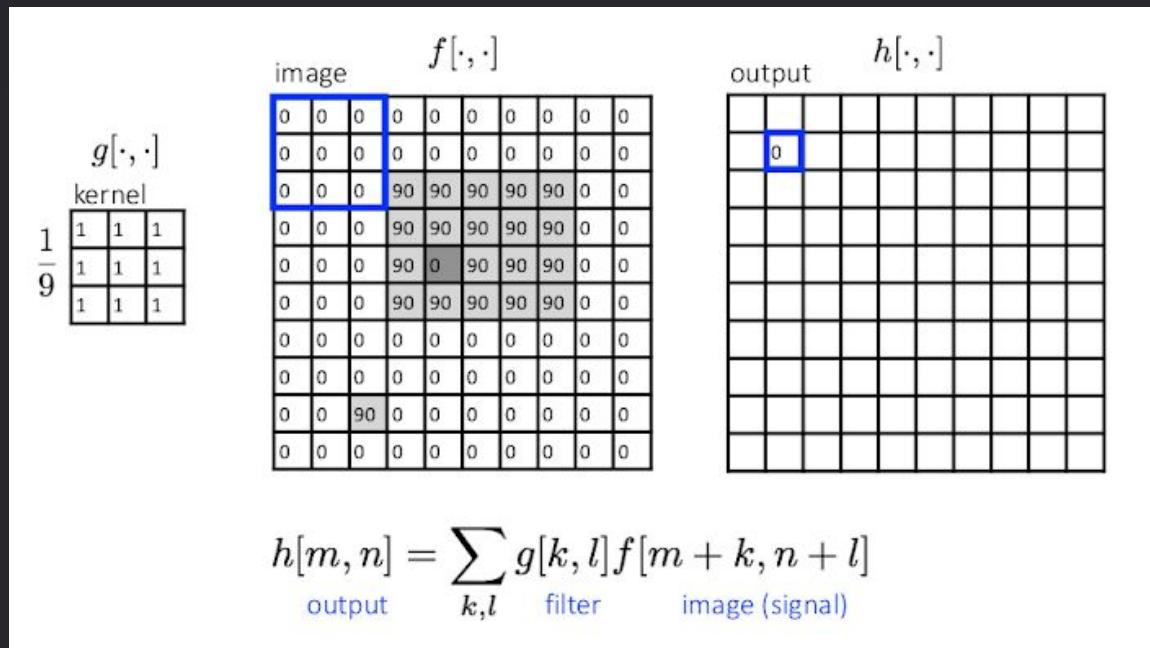


Filtering |

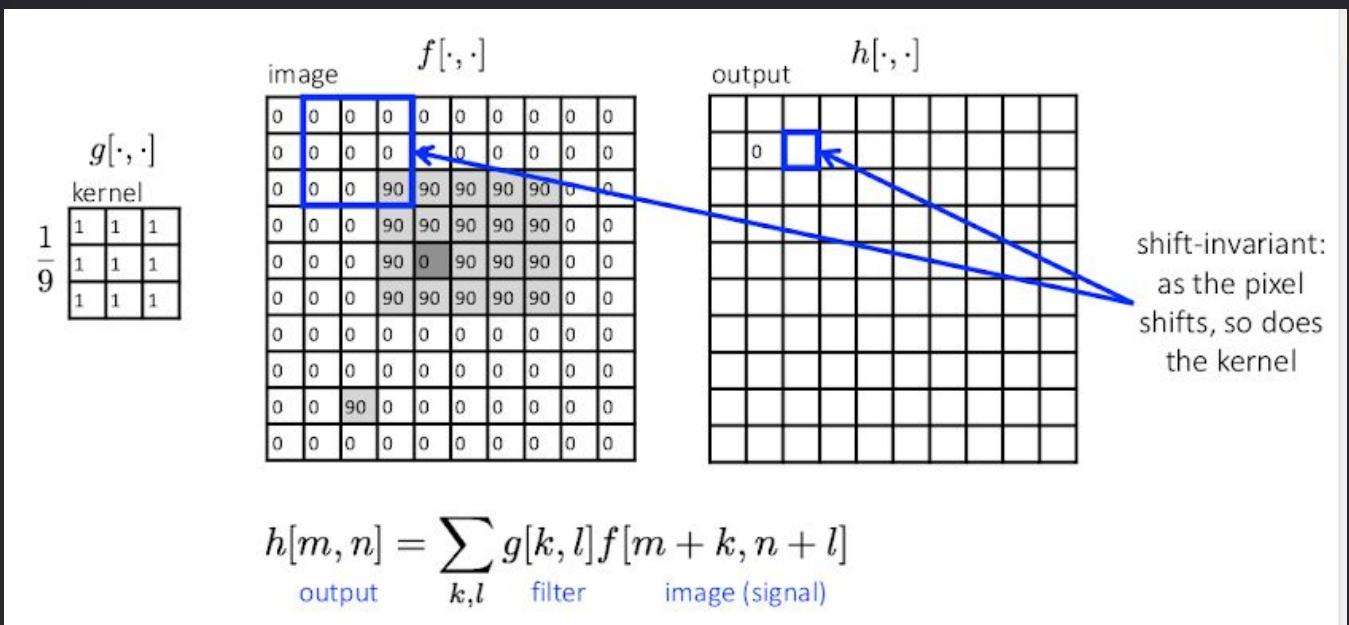
Sample: Box filter - Moving Average in 2D



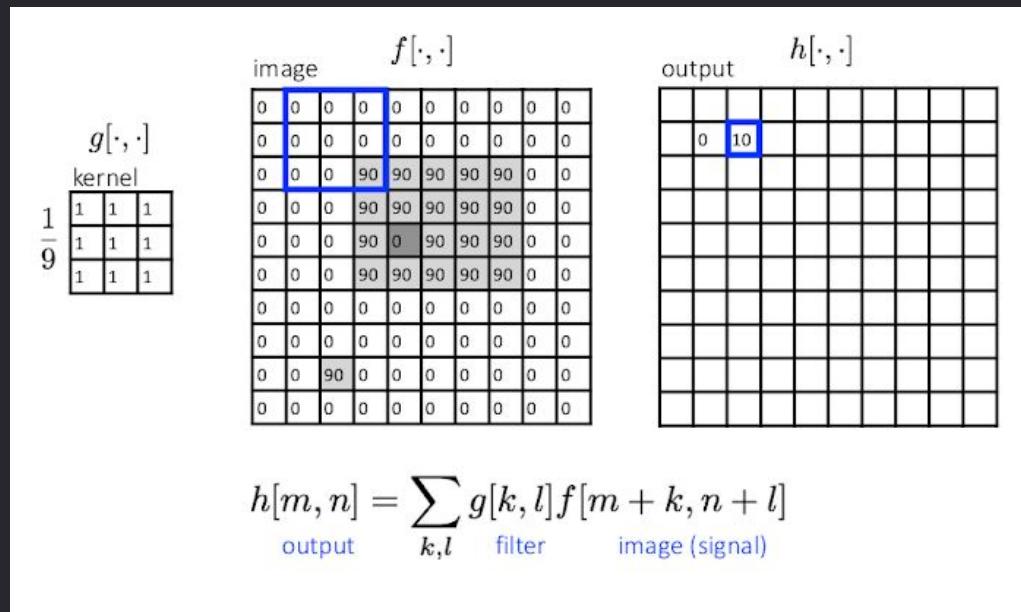
Sample: Box filter - Moving Average in 2D



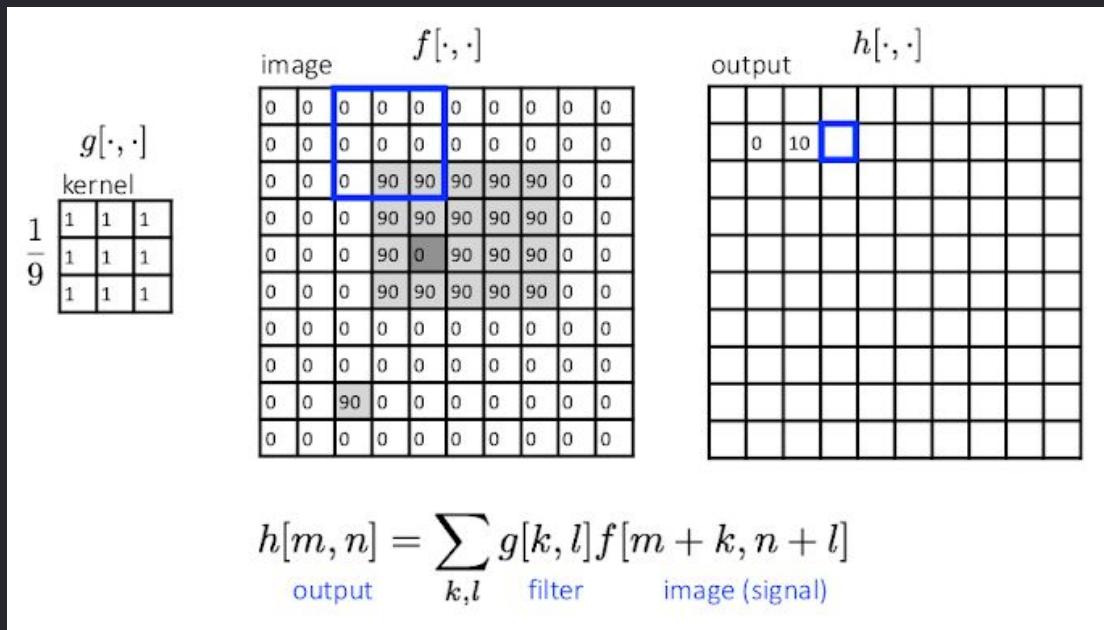
Sample: Box filter - Moving Average in 2D



Sample: Box filter - Moving Average in 2D

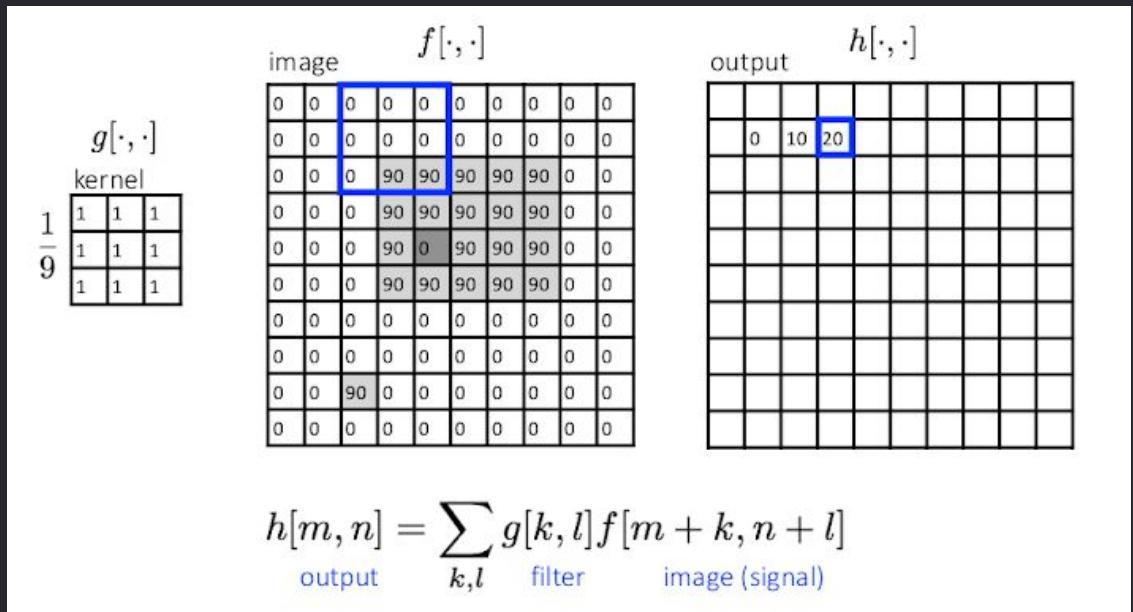


Sample: Box filter - Moving Average in 2D

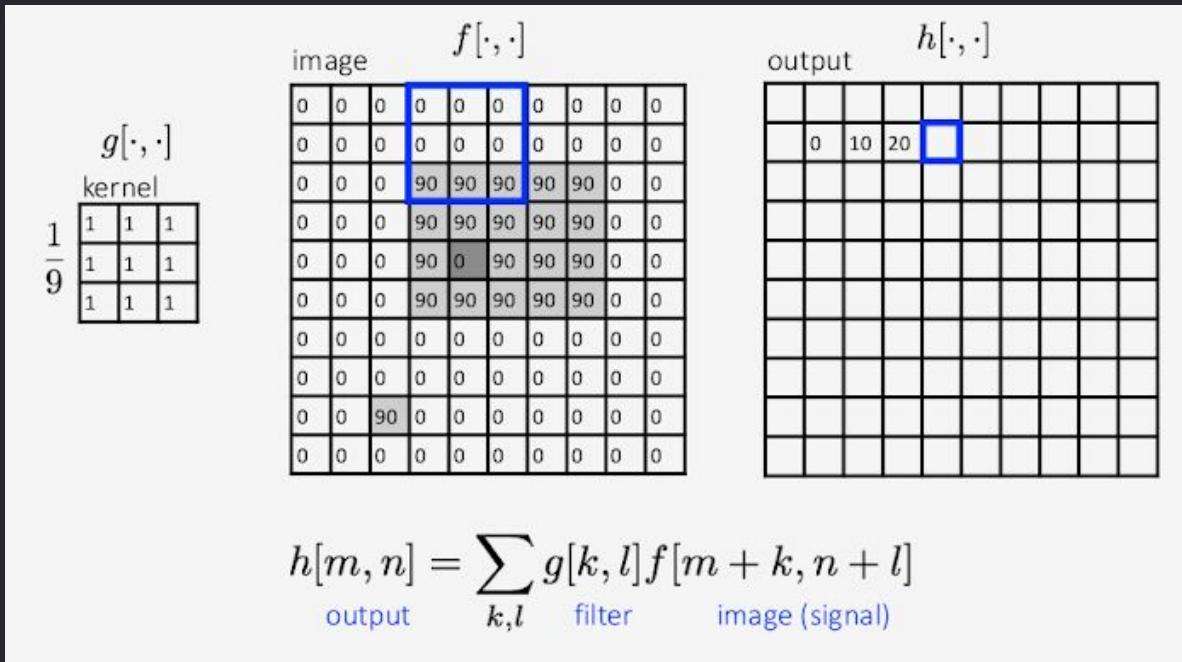


Filtering |

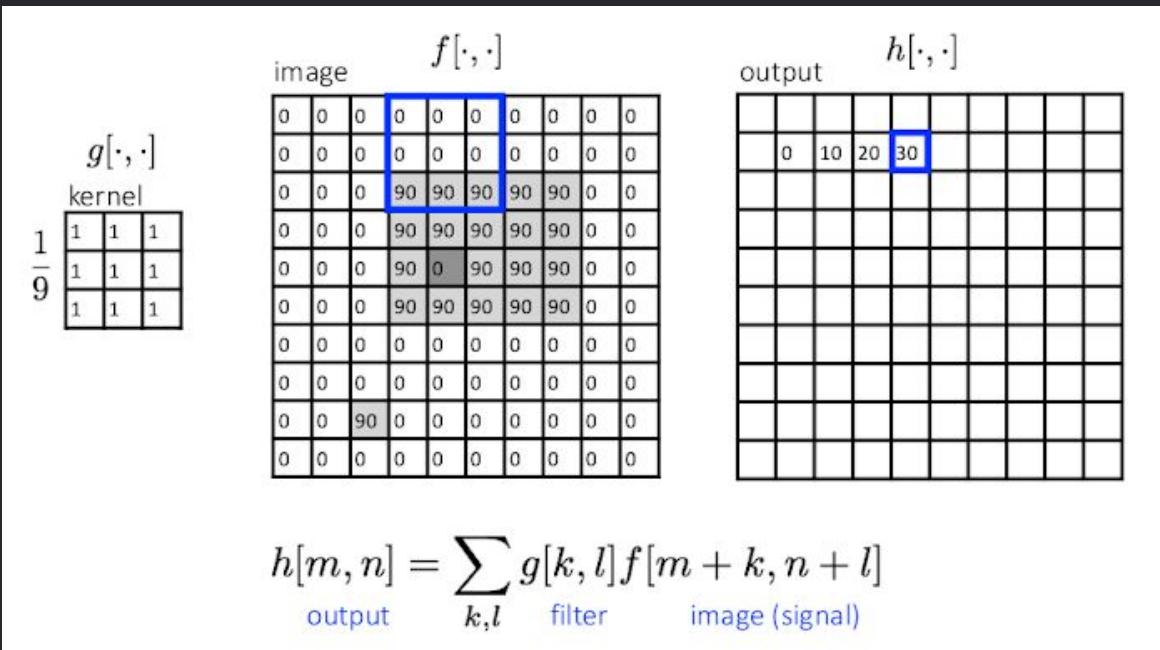
Sample: Box filter - Moving Average in 2D



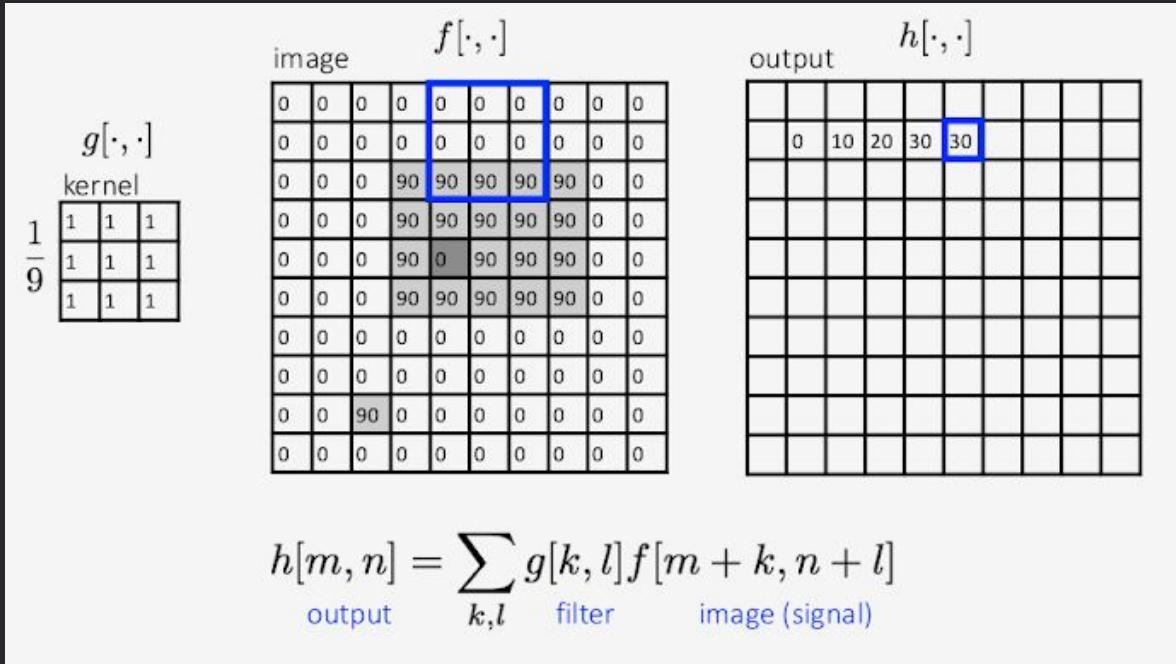
Sample: Box filter - Moving Average in 2D



Sample: Box filter - Moving Average in 2D

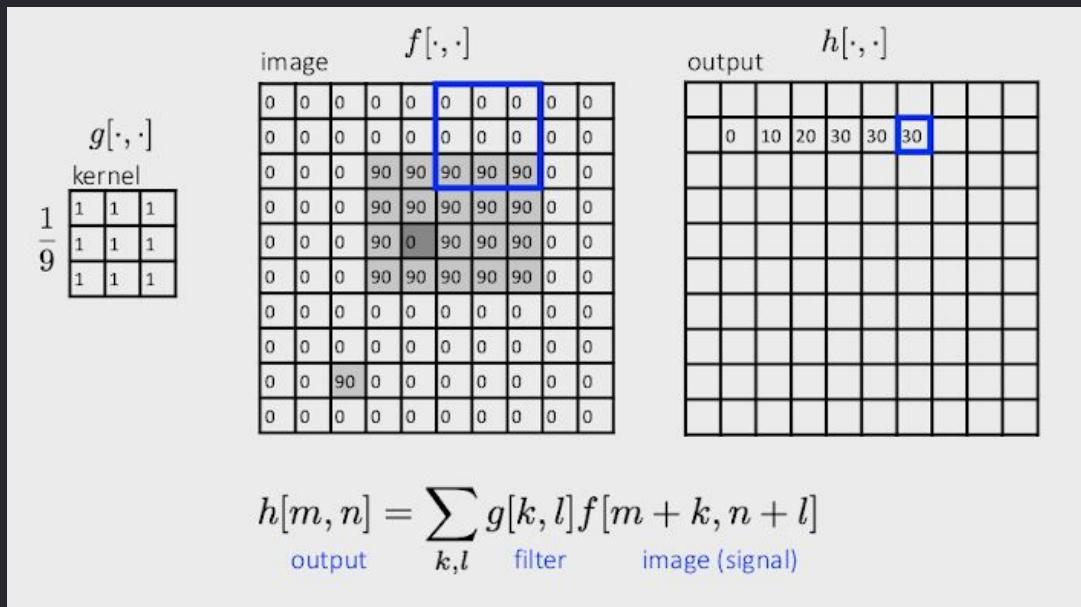


Sample: Box filter - Moving Average in 2D



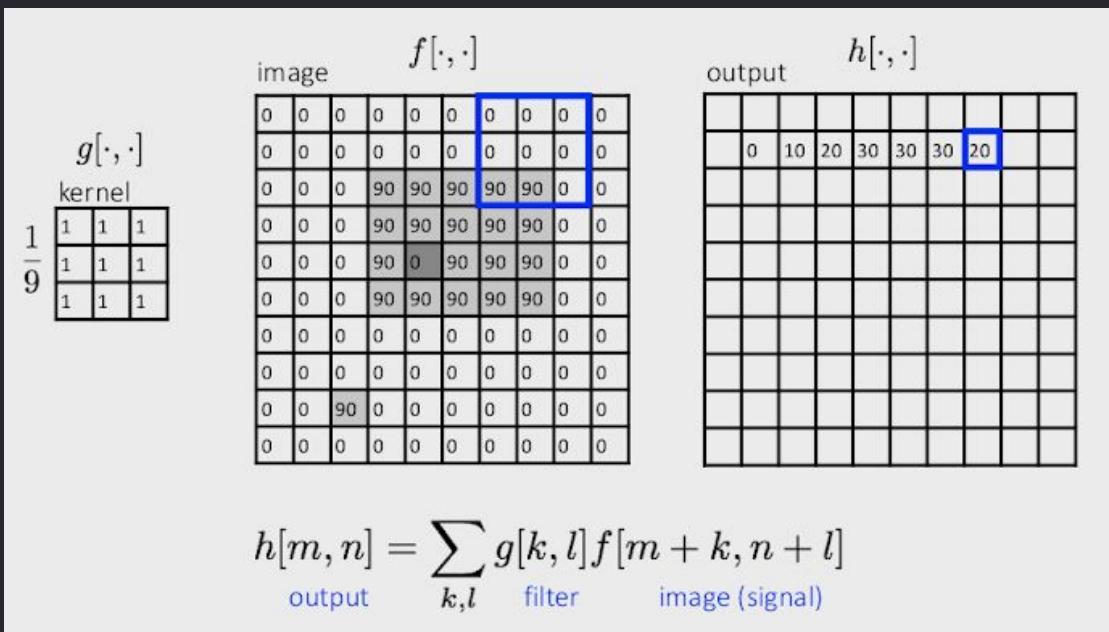
Filtering |

Sample: Box filter - Moving Average in 2D

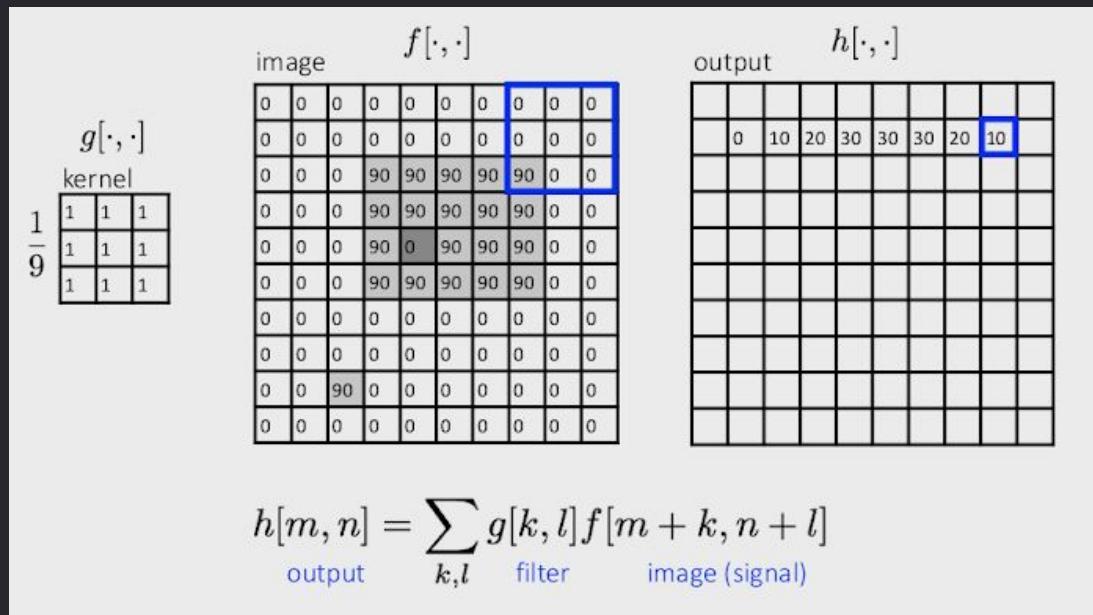


Filtering |

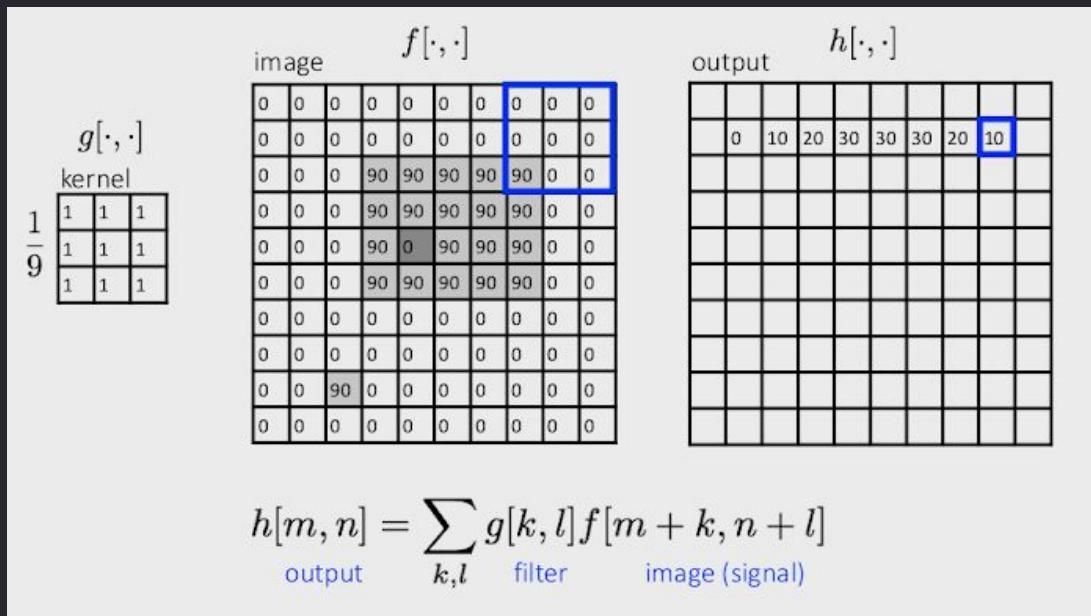
Sample: Box filter - Moving Average in 2D



Sample: Box filter - Moving Average in 2D



Sample: Box filter - Moving Average in 2D



Sample: Box filter - Moving Average in 2D

continuing for each line ...

Sample: Box filter - Moving Average in 2D

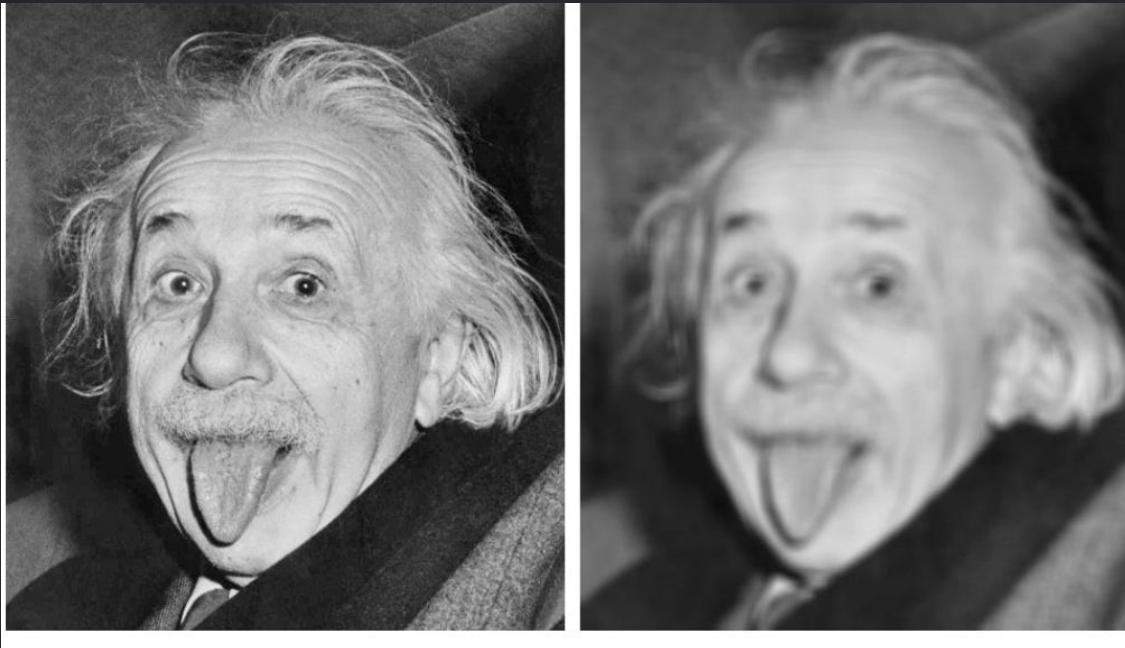
... and the result is

image	$f[\cdot, \cdot]$	$h[\cdot, \cdot]$
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 10 20 30 30 30 20 10
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 20 40 60 60 60 40 20
0 0 0 90 90 90 90 90 0 0	0 0 0 90 90 90 90 90 0 0	0 30 50 80 80 90 60 30
0 0 0 90 90 90 90 90 0 0	0 0 0 90 0 90 90 90 0 0	0 30 50 80 80 90 60 30
0 0 0 90 0 90 90 90 0 0	0 0 0 90 0 90 90 90 0 0	0 20 30 50 50 60 40 20
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 10 20 30 30 30 20 10
0 0 90 0 0 0 0 0 0 0	0 0 90 0 0 0 0 0 0 0	10 10 10 10 0 0 0 0
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	10 10 10 10 0 0 0 0

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output filter image (signal)

Sample: Applying Box filter result



Sample: Applying Box filter result



Sample: Applying Box filter result



Filtering |

Sample: Applying Box filter result



Sample: Image segmentation

$$g[n, m] = \begin{cases} 255, & f[n, m] > 100 \\ 0, & \text{otherwise.} \end{cases}$$

Sample: Image segmentation



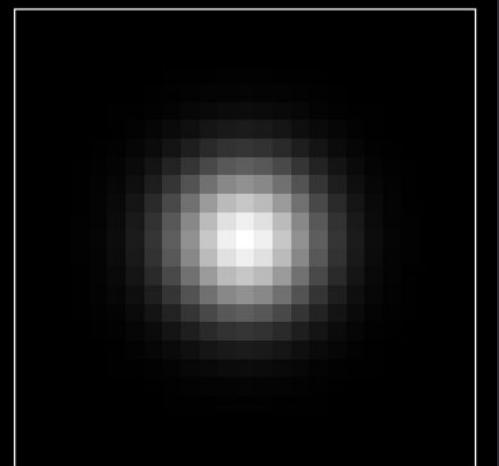
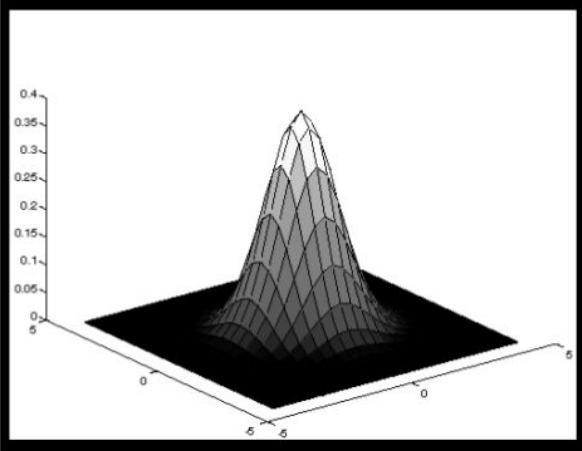
Sample: Gaussian filter

nearest neighboring pixels have the most influence

Sample: Gaussian filter

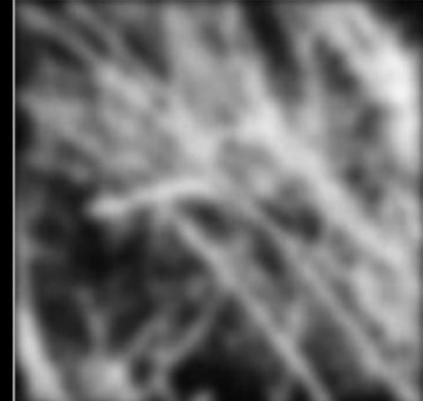
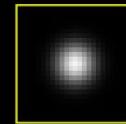
an Isotropic Gaussian

Sample: Gaussian filter



Sample: Gaussian filter

Smoothing with a Gaussian



Sample: Gaussian filter

Smoothing with not a Gaussian



Resources & References |

1. <https://www.forbes.com/sites/markhughes/2016/01/28/how-to-save-the-terminator-franchise/#692400624f72>
2. <https://www.udacity.com/course/introduction-to-computer-vision--ud810>

Thank you for your time & attention!