Perception CS4501 - Robotics for Software Engineers

By Carl Hildebrandt



Self-driving Case Study









Sense

Perceive







Plan

Act

Drone



Sense

Perceive

Plan







Husky









Self Driving Car









Perseverance























What is an image to a robot?

Question

Image Data

ROS: sensor_msgs/Image

sensor_msgs/Image Message





47, 851,	[40, 4	8, 861,	[49, 49,	671, [56	, 50, 6	81, [53,	53, 71	, [53]	52, 7	21, [50,	49, 69	, [49	44, 5	5], [48,	43, 64], [46,	43, 591	[46]	42, 611	. [47, 4	4, 591,	[51, 4	8, 631,	159, 4	8, 62)	, 147, s	45, 571	, 145,	43, 551,
35, 49],	. [43, 3	7, 48],	[46, 41,	50], [43	, 39, 4	5], [44,	43, 46)	, [44,	, 43, 4	7], [48,	43, 44	ļ, (54,	, 48, 4	9], (8 0 ,	55, 54], [62,	57, 58]	, [72, 1	83, 59].	, [83, 7	4, 71],	[98, 9	0, 83],	[117,	107, 9	7], [13]	5, 123,	111],	[147, 13
130], [1	172, 159	, 137],	[180, 16	6, 143],	[193, 1]	77, 151],	[195,	182, 1	L59], [199, 184	, 165],	1285,	189, 1	76], [21	1, 194,	181),	218, 20	0, 183]	, [219,	200, 15	5], [224	l, 284,	187],	228, 2	85, 19	8], [23]	2, 207,	194],	235, 26
1991, [2	547, 228	, 286],	[248, 22	2, 190],	[248, 2	19, 190],	, [248,	222_{r} 1	190], [250, 224	, 200],	251,	226, 2	20], [25	1, 226,	200],	251, 22	4, 284]	, [251,	224, 28	4], [25]	, 225,	2051,	253, 2	26, 26	5], [25]	3, 225,	2051,	252, 22
201 , [2	51, 225	, 201],	[254, 22	8, 284],	[255, 2	29, 205],	. [255,	229, 2	225],	255, 228	, 202),	255,	226, 2	21), [25	5, 226,	$199]_{r}$	254, 22	5, 198	, 252,	223, 19	6), [25]	5, 223,	196),	252, 2	22, 19	3], [25]	2, 222,	193],	251, 22
190], [2	82, 221	, 190 <u>]</u> ,	[252, 22	1, 158],	[251, 2	21, 186],	, [254,	228, 1	L84], [250, 216	, 150],	1248,	215, 1	76], [24	6, 213,	174],	245, 21	θ_r [172]	, [248,	212, 17	4], [248	3, 212,	$176]_{r}$	248, 2	12, 17	6], [250	ð, 213,	175],	[249, 21
1751, [2	547, 218	, 176l,	[248, 21	8, 170],	[248, Z	18, 170],	, [248,	280, 1	176], [247, 209	, 179],	248,	200, 1	26], [24	8, 200,	$170]_{\pi}$	243, 28	7_{r} [177]	, 1242,	286, 17	6], [234	, 20 <u>5</u> ,	1771_{r}	249, 2	87, 18	1], [238	9, 209,	182],	241, 21
190], [2	33, 216	, 194],	(229, 20	8, 193],	[230, 20	07, 199],	[223]	264, 1	196], [225, 205	, 226),	222,	203, 1	98), [22]	3, 284,	201),	222, 28	8, 202)	, 222,	287, 22	4], [224	, 280,	226],	220, 2	89, 21	1], [22]	2, 215,	214],	212, 21
219], [2	18, 212	, 2231,	(214, 21	6, 2241,	[215] 2	17, 228],	[212,	214, 2	225], [212, 212	238],	1287,	208, 2	22], [28	5, 205,	223],	281, 28	1, 215)	, [282,	199, 21	4], [28]	, 284,	214],	281, 2	81, 20	7], [19]	9, 201,	201],	283, 26
283], [2	18, 202	, 280],	[212, 20	2, 282],	[215, 2	83, 281],	[216,	266, 1	1901, 1	228, 282	, 191],	222,	284, 1	87], [23	8, 286,	186],	234, 210	0, 180	248,	212, 18	0), 1235	, 213,	189],	243, 2	15, 19	1], [24	8, 212,	188),	242, 21
159], [2	49, 218	, 191],	[247] 21	7, 158],	[248], 2	16, 157],	[248]	216, 1	157], [248, 218	158),	247.	216, 1	55], [24	8, 217,	155],	245, 21	8, 153)	248.	218, 15	3], [250	. 221,	182],	252, 2	21, 18	2], [25]	2, 220,	179].	253, 21
18L1, [2	52, 218	, 1821,	[252], 21	8, 1521,	[251, 2	17, 1811,	[252]	216, 1	L50], [252, 216	158],	1252.	216, 1	58], [25.	3, 217,	181),	254, 21	8, 182)	255.	219, 18	3], [25]	. 218,	184],	255, 2	29, 18	6], [25]	5, 221,	188].	255, 22
192] [2	55, 221	. 191].	[254] 22	1, 192].	[255] 2	24, 190].	[255]	223, 1	(94], [255, 222	192],	1255	222, 1	92], [25	5, 222,	189],	255, 22	3, 180)	. 1255.	223, 18	7], [25]	224,	180],	255, 2	23, 18	7], [25/	4, 225,	188],	255, 22
159 . [2	54. 224	1591	[254] 22	3. 1981.	254. 2	23. 1901.	255	224. 1	1591. 1	255. 223	158].	255.	223. 1	581. [25	5. 223.	158].	254. 22	2. 155]	255.	223. 18	71. [25]	. 223.	1871.	255. 2	22. 18	61. 25	5. 223.	1841.	255. 22
18LL 12	55. 223	1501.	1255. 22	3. 1501.	1255. 2	21. 1781.	1255	221. 1	1781. I	255. 222	1791.	1255.	223. 1	581, 125	5. 223.	1581.	255. 22	3. 1581	1255.	226. 18	21. 125	228.	1841.	255. 2	20, 18	51. 125	5. 228.	1841.	255. 22
1071. 12	55, 220	. 1071.	[255. 22	7. 1961.	[254] 2	26. 1151.	. D255.	227. 1	U61. I	255. 220	1871.	254	228. 1	w1. 125	2. 226.	1461-	255, 22	1, 1911	. 1255.	228. 19	11. 1252	. 220.	1911.	255. 2	29, 19	21. [25]	1. 229.	1921.	255, 22
1911. [2	55. 229	1921	[254. 22	8. 1581	[255. 2	26. 187]	[253	228. 1	1881 I	253. 227	187]	254	228 1	58]. [25	4. 220	189]	255. 22	6. 130]	254.	227. 19	61. [25]	. 229	1921	254. 2	27. 19	81. [25]	4. 231.	1931	255. 22
1951. 12	53, 226	1891	1252. 22	0. 1911.	1255. 2	20. 1921.	1254	238. 1	1941. 1	253, 220	1931	1255	220. 1	931. 125	4. 228	1921.	1255. 22	0. 1931	1255.	238. 19	41. 125	. 229.	1951	1255. 2	20, 19	51, 125	5. 227.	1931.	1255. 22
1041 12	85 704	1011	[255. 22	2 1001	F255 2	25, 100],	F 2555	2221 1	uu1 1	255, 223	1.001	1355	322 1	nul los	 52.0 	1011	1355. 32	7 1021	1956	126 10	11 125	1 125	1001	254 0	20, 10	11 (25)	1, 125	1981	2551 15
1044 [3	1997 - 221 1988 - 221	1001	[354 37	1 1001	Fore 2	20, 1901, 21, 1921	F 1000	2239 1	10010 1	2559 22.5	10017	12000	110 1	0917 180 0917 180	2 21 K	1911	16000 cc. 1960 - 91.	A 1381	12001	220/ 13	21, 1250	1 2207	1401	inen n	12 13	al 1963	1 112	1011	12001 22
1711 12	NG 200	1701	1245 24	8 1701	1242 2	85 (701	1244	200 1	691	245 200	1601	1245	21.6 1	671 124	2, 200,	1451	1241 24	5 1251	1241	285 14	51 124	, 215,	1641	1345 3	95 15	71 12.47	1, 240,	1601	12.45 24
171 6	2003, 2003 2015 - 2004	1221	(245) 26	1 1201	[245, 2 [248]	an 1991,	En an	200, 1	100), 1	245, 200	1120	12421	21.0, 1	11) (24 11) (28	1 115	1021	242, 20.	5, 100 J	1242,	114 12	5), (24) 5) (29)	, 200,	1741	242, 2	10, 10	k) [24]	5, 200,	10012	2242, 20
TRUE C	10. 2009	1001	[216 20	1 1001	[2240], 21	as 150],	[240]	200, 1	1001	2479 211	1061	230	21.4	2017 120 061 123	19 210y 8 210y	1031	12907 - 219 1388 - 339	0, 1791 8 1013	1240	210, 17	937 1231 91 (129)	1 2117	1043	inan a	10, 17	21 (22) 21 (22)	5 200,	1021	(223) 20 (226) 22
10041 12	28, 200	, 1001), DBG1	1225, 20	2, 1041, . 0, 0041	1239, 20	65, 198), 64 - 2241	1240,	206, 1	197], I.	245, 212	2201	1244,	214, 1	20), (24 Min 147	Dy 211, Dy 100	1001	1250, 22	6, 191). A 1671	12473	217, 10	6), (240 41 - 1450	, 220, . 305	12012	1453 2	20, 17,	3], [240 31 - 14-40	5, 221, 5 - 263	12715	14.22 . 21
2041), (2 2041), (2	23, 203	20012	. (2104), 200 Family 200	oy zitiy.	(2005) IS	or, zirj, an tanl	. (195). Fair	100, 2	(12), (104, 100	, 200), 	12149	110, 2	er) (an	e, 100,	12019	104, 10	0, 19() A 19()	12021	100, 20	9), (138 2), (138	, 100,	zesj,	(100) J	05, 29. NE 20.	5), (194) 2), (194)	5, 100,	12914	(1.35) 15 (1.84) 15
1991 y 11 1991 y 11	120, 145	9 1901» 	(122, 14 (122, 17	ty 1921y -	LILY, L	30, 1091, 20, 1091,	Elloy Faco	136, 1	1001, 1	117 - 1321	10012	1120 -	130, 1	801, 113 821 - 128	0, 198, 5. 198,	10117	1132 - 195 Image - 195	ny 1841. Ny fivon	 1101 1201 	10179 110	017 1098 2017 1098	1027	10217	1159, J Jacob J	00, 19	21, 110. 11 19.00	1, 100,	1901,	1101, 10
2001, [1	186, 186	, 1991). 	(162, 17	(, 1961). A 1961	1155, 1	73, 192], 73, 192],	1159,	176, 1	196], I 1961 - I	155, 166	, 150],	1150,	104, 1	57], [15 561 - 146	9, 105, 4 175,	10819	1259, 18	5, 168) A 1661	, 1155, 1464	104, 15	(, 15 , 01 140)	, 105,	1081,	1157, 1	100, 180 100, 200	8], [10:	5, 170,	1281,	1104, 17
1041_{2} (1	178, 177	, 1961,	(1/1, 1)	9, 1961,	11/0, 1	$(i_1, 102),$	1165,	1/5, 1	1961, 1	165, 173	, 156),	1005.	172, 1	56), (16 30), (16	4, 1 <i>13</i> ,	15719	1165, 174	4, 168) - 168)	, 1164,	172, 10	9], [16: 9], [16:	5, 1(1,	1081,	162, 1	69, 18	8, 15	5, 169,	10/1,	160, 10
1911, 11	63, 173	, 191,	1163, 17	3, 1981,	1168, 1	73, 1091,	1168,	173_{e} 1	1091, 1	168, 173	, 1091,	1163.	173_{r} 1	90], 116	1, 171,	1001,	1158, 16	6, 1891	, 1359,	168, 14	01, 1150	5, 164,	1871	1150, 1	64, 18	(1, 115)	1, 162,	1451	1154, 10
1571, [1	162, 109	, 1581,	(161, 17	8, 1541,	164, 1	73, 1571,	167,	174, 1	1591, 1	170, 177	, 192),	1173,	182, 1	25), 117	7, 187,	225	182, 19	2, 218)	, 184,	195, 21	71, 118,	, 197,	223 ,	182, 1	97, 22	3, 18	2, 197,	224 ,	182, 19
226 [, []	184, 196	, 2241,	[182, 19	4, 2221,	181, 1	88, 2151,	182,	186, 2	211], [184, 185	, 205],	185,	185, 2	21], [19	4, 188,	1991,	281, 19.	2, 195]	, 284,	192, 19	2], [28]	, 191,	187],	285, 1	93, 18	1, 28,	2, 190,	178,	287, 19
1241, [2	80, 129	, 188],	[289, 18	6, 164],	[289, 10	80, 161],	, [286,	198, 1	(99], [289, 191	, 168],	1211,	191, 1	88], [21	3, 109,	1591,	217, 19	1, 161)	, 1223,	191, 10	2], [220	5, 198,	1611	1222, 1	99, 16	111, [[]	a, a, a	1, [0,	e, e], [
10, 0,	6], 10,	0, 01,	10, 0, 0	1, 10, 0,	91, 19	, 0, 01,	13, 9,	01, 10	8, 8, 8	1, 18, 8	, ej, je	8, 8, 8	1, 10,	8, 81,	10, 8,	61, 10,	0, 01,	10, 0,	01, 10,	0, 01,	10, 0, 0	N, 10,	a, o),	18, 8,	91, 1	8, 8, 8	. 18, 1	8, 81,	10, 0, 0
91, [8,	a, ə),	16, 6,	81, 16,	8, 8], 18	, 6, 6]	, 10, 8,	6], [0,	6, 61	, (0,	0, 01, 1	ə, ə, ə	, la,	o, ol,	0, 0,	21, 18,	0, 01,	0, 0,	8], [8,	8, 81,	8, 8,	8], [0,	0, 01,	0, 0,	61, 16	, 0, 0	1, 10, 1	o, ol,	10, 0,	01, 10,
. 0, 8],	[8, 6,	6] [9	a, a],	[8, 8, 8]	, [0, 0	, a], [a,	. a, o],	18, 8	e, el.	18, 8, 8	1, 10, 0	ų, 61,	10, 8,	8], [0,	0, 0],	[0, 0,	[0], [0]	0, 0],	[0, 0,	o], [a,	0, 0],	[0, 0,	a), [0,	, 8, 8)	. 10, 1	8, 81, 1	[0, 0, 1	0), (e,	0, 0],
, [0, 3,	31, 18	, 8, 8]	, 18, 8,	el, le, e	, 0], [0, 0, 01,	[0, 0,	01, I	0, 0,	əl, [ə,	a, al, I	0, 0,	01, [3	, 0, 0],	[θ, θ,	0], [0	, 8, 8],	18, 8,	8], [8	, 0, 0],	10, 0,	01, [0	, 0, 01,	, 10, 0	, el,	10, 0, 0	al, IØ,	0, 01,	[θ, Θ,
, 01, 13	0, 0, 01	, [0, 0	, 01, 10,	0, 01, 1	a, a, a	1, 10, 0,	al, 16	, 6, 6	31, 10,	0, 01,	10, 0, 0	31, 10,	0, 01	, [0, 0,	et, të	, 0, 01	, 10, 0,	01, 10	, 0, 01,	, le, o,	01, 10,	0, 01	, 10, 0,	, 01, I	θ, θ,	81, 10,	0, 01,	10, 0,	01, 10,
8, 8, 8]	, [8, 8	, 8], [8, 8, 8],	0, 8, 8], [8, 1	8, 0], [2	5, 8, Ø]	, [ð,	a, a],	[0, 0,	a], [a,	0, 0]	[8, 9	, a], [e	, 8, 8]	, 10, 0	, 0], [0	, 0, 0]	, [D, U	, 0], [e	, e, e],	[0, 0	, e], [(e, e, e], [0,	0, 0],	[0, a,	e], [8	, 0, 0],
1, 10, 1	a, al, (0, 0, 0	1, [0, 0,	ð], [0,	a, e],	[0, 0, 0]	1, 18, 6	J, B],	10, 0,	0], [0,	0, 0],	10, 0,	, 01, (ə, ə, el	, [0, 0	, 01, D	0, 0, 0l	, [0, 0	, 0], []	8, 8, 0]	, 18, 8,	. 0], [8, 8, 8	1, 10,	8, 8],	10, 0,	0], [0	, 0, 0]	, [0, 0,
e, el, l	10, 0, 0	1, 10,	e, el, le	, 0, 01,	[ð, ð,)	01, [2, 4	8, 81, I	0, 0,	əl, lə	, ə, əl,	[0, 0,	ēl, 16	3, 8, 3	1, 10, 0	, 01, I	8, 8, 8	1, 10, 8	, 01, L	2, 0, 0	1, 12, 0	, el, D	ε, ο, ο	1, 10, 6	0, 0I,	10, 0,	01, 12,	, 0, 01,	, 12, 8	, 01, 12
[8, 8, 8	3], [8, I	ə, ə].	[8, 8, 8]	. [8, 8,	a], [e,	8, 8], [8, 8, 8	1], [8,	. 8, 8]	, [0, 0,	8], [0,	0, 0]	, [a,	ə, ə], [2, e, e], [2, 4	a, a], [2, 0, 0	[1, 10, 1]	8, 0], [0, 0, 0)	. 10,	0, 0], 1	(e, e,	8], [0	, e, e],	, [e, e	, 0], [e, e, e]
01, 12,	e, el,	10, 6,	el, 15, e	, 0l, îð,	ə, əl,	[2, 0, 0	M, [2,	ə, əl,	, [2, 0	, 0], [2	, 0, 0],	, 12, 0	8, 8],	12, 8, 8	1, 12,	8, 8],	0, 0, 0	l, [0, 1	o, ol, i	[0, 0, 0	l, [0, (5, 0l,	[0, 0, 4	ol, f0,	0, 0]	, 18, 8,	, 0], 0	8, 8, 8	1, 12, 8
ə, əl,	12, 0,	01, 12,	8, 81, 1	2, 0, 01,	12, 0,	01, 12,	8, 81,	12, 0,	, 01, I	2, 0, 01	, 12, 0,	, 01, I	0, 0,	81, 10, H	a, al,	10, 0, I	31, IB, K	0, 01,	10_{T} 0_{T} 0_{T}	81, 18,	8, 81,	2, 8,	81, 12,	0, 01,	12, 0	, 01, D	2, 0, 0	1, 12,	0, 01, 1
$[3i, 1]_{a}$	1], [3,	1, 1],	[3, 1, 1], [3, 1,	1], [2	, ə, ə],	[2, 0,	ð], [2	2, 8, 8	[1, [2, 8]	, 0], [2	2, 8, 6	(], [2 _a	8, 8],	2, 0,	0], [3]	J, J],	[3, 1,	1], [3,	1, 1],	[4, 1, 1	1, [6,	2, 1],	[4, 2,	1], [$7_{r} = 3_{r} = 2$], [10,	6, 5],	[11, 2,
, 11, 18	M, [25,	17, 17	1, [31, 2	3, 231, f	35, 27,	271, [44	4, 36, 3	07] , [4	14, 38,	39], [4	5, 43, 4	451, fs	50, 44,	491, 15	1, 47,	521, 15.	3, 52, 5	61, 159	, 37, 63	31, 169,	65, 71	1, 193,	76, 82	1, 188,	84, 9	01, 190,	, 86, 93	21, 188	, 84, 96
, 79, 83	51, 180,	76, 82	1, 179, 7	5, 81I, I	78, 74,	801, 177	7, 73, 7	91, 17	79, 75,	811, [8	3, 76, 8	831, 18	32, 75,	821, 18	4, 76,	831, 18	5, 77, 8	41, 187	, 80, 83	51, 189,	82, 87	, 198,	83, 88	1, 191,	84, 8	91, 192,	, 85, 9	01, 191	, 84, 89
, 82, 82	[], [93.	85, 91], [93, 8	6, 91], [92, 85,	98], [93	5, 58, 9	6], [9	45, 98,	85], [9	9, 93, 1	46], [1	120, 94	, 99], [1641, 98	, 103],	[102, 9	6, 101]	, [99, 1	97, 99],	[97, -9]	, <u>99</u>],	[103, 3	99, 165], [100	2, 95, 1	183], [101, 97	, 102],
2, 97, 1	1881, 19	1, 55,	951, 189,	84, 931,	198, 8	5, 941, 1	191, 83,	931.	194, 8	8, 56],	91, 84,	91].	192, 8	5, 94],	196, 92	, 981,	199, 98,	1011,	100, 90	8, 102l,	198, 99	i, 1821	, 197, 3	95, 101	1, [98]	, 95, 10	24], [1	80, 97,	186], [
6, 1021,	196, 9	4, 1001	, 197, 94	, 1031, i	98, 95,	1841, 11	182, 99,	1651,	198,	95, 1041	, 195, 9	2, 100	1, 195	, 91, 18	31, 195	91, 1	031, 197	, 93, 1	051, 191	7, 93, 1	851, 194	, 90,	1021, 18	89, 86,	1011,	188, 8	5, 1011	, 184,	82, 1011
77, 98]	, [84,	51, 183	[, [85, 8	1, 186],	[78, 74	, 99], [)	12, 69,	91], [[71, 69	, 91], [12, 71,	91], [[77, 72	, 82], [85, 84,	94], [94, 93,	102], [113, 110	8, 121],	123,	25, 13	3], [112	5, 118,	126],	1965,	188, 11	9], [18	9, 110,
1, 199,	99, 117	1, 193,	93, 1891	, [91, 91	, 107l,	[94, 94,	1121,	194, 9	14, 112	1, 199.	8. 114	, [11]	5, 115,	129], [118, 11	9, 139]	, [117,	128, 14	1], [119	5, 128,	141], []	17, 12	3, 1461,	, [117 ,	128,	148], []	124, 13.	3, 160]	F128,
. 1711.	1135, 1	45, 179	l, 1135,	144, 1781	, [135],	145, 175	91, 1134	1, 143.	1771,	1134, 1	45, 177	i, 1134	4, 146,	1741, 1	138, 15	3, 1791	1145,	161, 18	41, 114;	2, 159,	1801, 13	36, 15	3, 1741,	, 1133,	148,	1741, 13	133, 14	6, 1781	1133,
181].	[135, 1	47, 183	1. [137,	150, 182]	, [141,	151, 181	1], [143	, 151	181],	[146, 1	56, 185]	. [15)	1. 158,	188], [158, 15	9, 192]	[149]	158, 19	[1], [14]	7, 156,	193], [1	143, 15	2, 190].	. [140,	146,	187], [)	139, 14	5, 186]	[136,
. 1811.	[126. 1	38, 178	1. [128.	138, 178	[126.	139, 17)	1. [125	. 138.	1761.	[124. 1	37. 175	. [119	. 128.	196]. [113. 12	6. 164]	[118.	124. 16	0]. [11]	3. 125.	161]. D	14. 12	6. 1621.	[113.	125.	161]. D	112. 12	4, 168]	. F185.
. 1541.	[101. 1	10, 153	1, 199, 1	89, 1491.	1182.	169, 1521	1. 1154.	112.	1521.	1101. 11	0, 1481	1100	185.	1471. 11	20, 185	1451.	1102. 1	69, 148	1, 199.	187. 14	41, 196	104.	1411. 19	96, 183	, 1381	197.	186, 13	91, 196	, 106, 1
[98, 18	6, 1321	. [97.	184, 1311	[188.1	85, 135	1. [181.	185, 13	H]. [1	184, 18	4, 1331.	[95, 10	0, 131	1. 189	. 93, 12	21. [85	. 89, 1	17], [85	. 87, 1	17], 18;	2, 84, 1	14]. [8]	. 84.	112]. D	\$2, 86,	1141.	[85, 8 [,]	5, 1141	178	29, 1921
. 67. 91	1. [69]	64, 95	1. [69. 6	4. 951 . 1	69, 64	981. [22	68. 9	01. IZ	16. 72	1011. [\$1.78	1041	185. 8	2. 1881.	[84. 4	1. 1061	[35. 8	4. 1181	. 187.	86. 1121	. 195	8, 117	1. [95]	96. 12	41. 19	6. 98.	1281. 0	96, 98,	1281.
6, 1191		3. 1671	. 180. 50	. 1841. 1	84, 82	1841. 18	96. 84	1661.	185.8	5, 1651	189. 80	1. 1.001	L [95]	94. 114	1. 1101	. 99. 1	181. 118	5. 105	1231	188.14	7. 1271	1183	107. 1	271 1	87. 14	5. 1271	[181	102. 1	231. 197
	[BL. 8		1105 10	3. 1081	F188. 0	7. 1221	[97. es	1. 1191	. [108	. 97. 12	31. [184	1. 97	1231.	187. 58	1271	[97] Q	3. 1221	188 8	6. 3161	. [85. 8	3. 1191	E83	78. 189	1. 182	79. 1	851. De	1.11	1101 1	89. 88
11.91	. [81, 8 [99, 166	1211					the second se	the second se	Contraction of the local division of the loc	A 11 1 1 1 1		1 A A		1 1 1 1 1 1 1		1				Fuel -		and the second se		and the second se					
[119], [190], 20	. [81, 8 99, 100 	, 121], [118	189. 1951	[118 1	28. 128	1. [130	181 72	ell. [1	13718	9, 1581	[132	10. 11	[a] = [a]	24. 139	1441	[114]	16. 1421	. [122	1000 11	881. fas	. 193	1311 1	93. 129	1271	[91	QR. 125	1. 108	101 1	241. Isr
11.9], [[99, 99] 4. 124]	(81, 8 99, 100 8, 114], (99, 1	(, 121], [118, 27, 127	189, 1251 1. [120	, [118, 1 129, 1321	28, 128	I, [129, 110, 121	131, 13 N. 197	91, [1 187	137, 13 1311	9, 1501, 193, 193	[137, 1]	139, 15 195 - 1	80], [1 184, 13	24, 129, 11, 196	1441, 185, J	[114, 1] 321, 19	16, 1401 3. 166	, [103, 1351 - J	108, 11 99. 107	331, [98 . 1361	, 102, 1 (95, 10)	131], [. 133]	93, 100	, 1271, 181 - 13	[91, 1 11, 19	98, 125 3, 142	1, 193, 1351	101, 1	241, [94 5. 1381
11.9], 99, 99 4, 124], 84 189	(99, 160 (99, 160 3, 114], (98, 1 142]	, 121], [118, 87, 127	189, 125] 189, 125] 1, 1101, 88, 141]	, [118, 1 109, 1321 104 - 70	20, 128 , 1101, 0 1421	1, [129, 110, 130	131, 13 M, 197,	9], [1 187, 1 114	137, 13 1311, 13 13, 197	9, 1501, 193, 103 1421	[137, 1 , 127], [101	139, 15 195, 1 15, 149	80], [1 184, 13 81 119	24, 129, 11, 196, 8 101	1441, 185, 1 1421	[114, 1 32], 19 192 18	16, 1401 8, 106, 1 8, 1412	, [103, 1351, h 1140	108, 13 99, 107, 105, 107	331, [98 , 1361, 81 - [198	, 182, 1 195, 183 188	131], [3, 133] 146] - 1	93, 100, , 193, 184, 214	, 1271, 101, 13 2, 1423	[91, 1 11, 19 [100	98, 125 3, 102, 188	1, 193, 1351, 1451 - P	101, 1 198, 18	241, [94 5, 1381, 6, 1481
11.9], [99, 99 4, 124], 84, 189, 84, 119	(181, 8 (99, 169 5, 114], (198, 1 (142], (142],	, 121], [110, 87, 127 [101, 1 [107, 1	189, 125] 189, 125] 1, [101, 88, 141], 18, 140]	, [118, 1 109, 132] [184, 18	20, 125 , [101, 9, 142] 8, 142]	<pre>1, [129, 110, 130 , [104, 1]104</pre>	131, 13 M, 197, M8, 143	91, [1 187, 1, [19	137, 13 1311, 13, 107 15, 114	9, 150], 193, 193 , 142], 151]	[137, 1 , 1271, [101, 10]	139, 15 195, 1 13, 143	80], [1 104, 13 8], [10 9], [10	24, 129, 11, 196, 8, 101, 8, 121	144], 185, 1 142], [159] -	[114, 1 32], 19 108, 10 118, 12	16, 140] 8, 106, 1 8, 141], 5, 159]	, [103, 1351, 1 [190, [119]	108, 11 99, 107, 185, 140 127 151	331, [98 , 1361, 3], [185 7], [185	, 182, 1 195, 183 , 188, 1 182	131], [3, 133] 146], [161]	93, 108, , 193, 184, 110	, 1271, 101, 13 0, 147] 0, 1821	[91, 9 11, 19 , [102	98, 125 3, 102, , 188, , 125	1, 193, 1351, 145], [150] (101, 1 198, 18 100, 18	24], [94 5, 138], 6, 143], 7, 157]
11.9], [[99, 99 4, 1241, 84, 189, 84, 118, 25, 132	<pre>[99, 181, 8 [99, 1896 3, 114], [98, 1 [142], [147], [152]</pre>	(, 121]), [110, 87, 127 [101, 1 [107, 1 [125]]	189, 1251 1, 1101, 88, 141], 13, 158], 35, 128]	, [118, 1 109, 1321 [104, 10 [102, 11 1125, 12	20, 128 , [101, 9, 142] 8, 147] 5, 169]	<pre>1, [129, 110, 130 , [104, 1 , [104, 1 , [104, 1 , [130, 1]</pre>	131, 13 M, 197, M8, 143 112, 149	91, [1 187, 0, [10 0, [10 0, [16	137, 13 1311, 13, 107 36, 114 30, 132	9, 150], 193, 193 , 142], , 151], , 168]	[137, 1 , 1271, [101, 10 [114, 12 [129, 12	139, 15 195, 1 13, 143 12, 159 12, 159 12, 162	80], [1 104, 13 8], [10 9], [11 7], [12	24, 129, 11, 196, 8, 101, 3, 121, 5, 139	1441, 185, 1 142], [158], [168], [[114, 1 321, 19 198, 19 118, 12 131, 12	16, 140] 5, 106, 1 5, 141], 5, 158], 9, 169]	, [103, 1351, 1 (100, [119, 1134	108, 11 99, 107, 105, 140 127, 15, 139, 127	881, [98 , 1361, 8], [185 7], [123 8], [123	, 182, 1 195, 183 , 185, 1 , 131, 1 182	131], [5, 133] 146], [161], [178]	93, 108, , 193, 1 184, 114 124, 13, 132, 124	, 1271, 101, 13 0, 147] 2, 162] 9, 1721	[91, 19 11, 19 , [102 , [121	98, 125 3, 102, , 188, , 129, 142	1, 193, 1351, 145], [1 159], [1 176], [1	101, 1 198, 10 100, 10 119, 12	24], [94 5, 138], 6, 143], 7, 157], 2, 175
11.9], [[99, 99 4, 1241, 84, 189, 84, 118, 25, 133, 39, 145	<pre>(81, 8 (90, 100) (90, 100) (, 114], (98, 1 (142], 1442], 147], 1631, 1763</pre>	[, 121], [118, 37, 127], [181, 1], [187, 1], [125, 1], [143, 2],	(1865, 18 189, 125] 1, [101, 88, 141], 13, 158], 35, 158], 45, 179]	, [118, 1 109, 1321 [184, 18 [182, 11 [125, 13 [145, 14	20, 128 , [101, 0, 142] 0, 147] 5, 168] 5, 1791	 [129, 110, 130, 130, 130, 130, 130, 130, 130	131, 13 N, 197, NB, 143 112, 149 139, 168	91, [1 187, 0, [18 0, [18 0, [18 1, [13	137, 13 1311, 13, 107 95, 114 95, 138	9, 150], 193, 183 , 142], , 151], , 168],	$\begin{bmatrix} 137, 1\\ 1271, \\ 101, 10\\ 114, 12\\ 1129, 13\\ 140, 24$ 140, 240, 240, 240, 240, 240, 240, 240, 2	139, 15 195, 1 13, 143 12, 159 37, 165 35, 195	80], [1 104, 13 8], [14 8], [11 7], [13 9], [13	24, 129, 11, 196, 8, 100, 5, 121, 8, 138, 8, 155	144], 185, 1 142], [158], [168], [185], [[114, 1 321, 19 198, 19 118, 12 131, 13	16, 140] 5, 166, 1 3, 141], 5, 158], 9, 169],	, [103, 135], F (140, [119, [134, [120]	108, 13 99, 107, 105, 143 127, 153 139, 170	881, [98 , 1361, 8], [145 7], [128 8], [132 8], [132	, 182, 195, 183, , 188, , 131, , 140,	131], [3, 133] 146], [161], [178], [198], [93, 109, , 193, 184, 114 124, 13, 132, 135	, 1271, 101, 13 0, 147] 2, 162] 9, 1721	[91, 19 11, 19 , [100 , [121 , [138	98, 125 3, 102, , 108, , 129, , 143, , 154	1, 193, 1351, 145], [1 159], [1 176], [1 182]	101, 1 198, 10 100, 10 119, 12 137, 14	241, [94 5, 1381, 6, 143], 7, 1571, 2, 1751, 3, 1921
11.9], [[99, 98 4, 1241, 84, 189, 84, 113, 25, 133, 39, 141, 48, 153	<pre>(81, 8 (92, 160), 114], (93, 114], (198, 1 , 142], (147], (1631, (176], 1892</pre>	(, 121), [118, 37, 127 [181, 1 [187, 1 [125, 1 [148, 1 [155, 2	(185, 18 139, 125] 1, [101, 88, 141], 13, 158], 35, 158], 45, 178], 55, 199]	, [118, 1 109, 1321 [184, 18 [182, 11 [125, 13 [148, 14 [151] 15	20, 125 , [101, 8, 142] 8, 147] 5, 168] 5, 168] 5, 178]	 [129, 110, 130, 130, 130, 130, 130, 130, 130	131, 13 M, 197, 108, 143 112, 149 139, 168 145, 177	(9), [1 , 187, (), [10 (), [10 (), [10 (), [13 (), [14 (), [14	137, 13 1311, 13, 107 95, 114 90, 138 15, 158	9, 150], 193, 103 , 142], , 151], , 168], , 181], 1921	[137, 1 , 127], [101, 10 [114, 12 [129, 13 [140, 19]	139, 15 195, 1 13, 143 12, 159 37, 165 35, 184 38, 199	80], [1 804, 13 8], [10 9], [11 7], [13 9], [15 8], [15	24, 129, 11, 196, 8, 101, 8, 121, 8, 138, 8, 156, 153	144], 165, 1 142], [158], [168], [185], [192]	[114, 1 321, 19 198, 19 118, 12 131, 13 151, 15 151, 15	16, 140] 5, 166, 5, 141], 5, 158], 9, 169], 5, 183], 5, 183],	, [103, 1351, F [140, [119, 1134, [149, [152]	108, 13 99, 107, 105, 143 127, 153 139, 170 153, 183	88], [98 , 135], 8], [145 7], [128 8], [132 8], [142 2], [142	, 182, 195, 183, 188, 131, 140, 151, 155	131], [5, 133] 146], [161], [179], [189], [192]	93, 100, , 193, 1 184, 114 124, 13, 132, 13; 150, 154	, 1271, 101, 13 0, 147] 2, 162] 9, 172] 4, 183]	[91, 19 11, 19 , [102 , [121 , [138 , [150 [150	98, 125 3, 102, 108, 129, 143, 154,), 01, 1351, 145], [: 159], [: 176], [: 182], [:	101, 1 198, 18 100, 18 119, 12 137, 14 149, 15	24], [94 5, 138], 6, 143], 7, 157], 2, 175], 3, 182], 1921

 57. 128], [128, 144, 216,

 188, 194], [244, 216,

 192], [250, 221,

 12, 174], [247, 211,

 13, 202], [213, 210,

 13, 203], [246, 264,

 14, 160], [233, 214,

 15, 203], [246, 264,

 15, 160], [243, 214,

 14, 160], [243, 214,

 15, 203], [254, 224,

 15, 189], [254, 224,

 12, 183], [254, 224,

 12, 183], [254, 224,

 12, 183], [254, 224,

 12, 183], [254, 224,

 12, 183], [254, 224,

 12, 183], [254, 224,

 13, 152], [253, 228,

 14, 193], [255, 228,

 19, 192], [253, 223,

 193, 1255, 228,

 194], [255, 228,

 195], [253, 228,

 196, 1255, 228,

 197, 192], [253, 228,

 198, 1761, 123, 120,

 193, 1625, 228,

 194, 1255, 228,

 195, 1248, 289,

 196, 1248, 289,

 197, 191, [125, 172,

 12, 133, 150,

 191, [148, 173, 108,

 191, [168, 173, 108,

 192, [12, 0, 01, 16, 00, 01, 16,

 1

Image Data









Image Width















Perception

"Perception refers to the ability of an autonomous system to collect information and extract relevant knowledge from the environment."

-Pendleton, Scott Drew, et al. "Perception, planning, control, and coordination for autonomous vehicles." Machines 5.1 (2017)

Perception

"Perception refers to the ability of an autonomous system to collect information and <u>extract relevant knowledge from the environment</u>."

-Pendleton, Scott Drew, et al. "Perception, planning, control, and coordination for autonomous vehicles." Machines 5.1 (2017)

Perception Examples

How: Processing sensor data to create a higher-level abstraction of the data







Traffic Light: Stop



Camera Sensor





Perception









Main Types of Perception

Classification



Object Detection

Interpretation



Perception

What: extract relevant knowledge from the environment Input: Raw data Output: Classication / Object Detection / Interpretation

How: ?





Perception Algorithms

Perception estimates the state of the environment

Image Processing Algorithms

An image is processed through parameterized transformations.

Key: We define this function



Machine Learning

Gather large amounts of data to learn or approximate the desired function.

Key: The computer learns this function



Example

Example

Input





Example

Input



Output

Image Processing Techniques

- Thresholding
- Color Filtering
- Blurring
- Smoothing
- Background subtraction
- Edge Detection
- Corner Detection
- Feature Matching
- Haar Cascade Object Detection
- •

Image Processing Techniques

- Thresholding
- Color Filtering
- Blurring

 \bullet

. . .

- Smoothing
- Background subtraction
- Edge Detection
- Corner Detection
- Feature Matching
- Haar Cascade Object Detection

Idea

Remove a range of colors from an image

Technical Implementation

Convert image into a format that makes selecting colors easy Look at each pixel, if it is not in your selected range remove it

HSV Image Format

HSV stands for Hue, Saturation, Value, and is a cylindrical color space. <u>Hue</u>: Are colors rotating around a central vertical axis Saturation: Defines the shade of the color from least saturated to most <u>Value</u>: Defines brightness from darkest to brightest

Code

13	# Convert from BGR to HSV color space
14	hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
15	
16	# Look for orange
17	lower_color = np.array([0, 80, 80])
18	upper_color = np.array([255, 255, 255])
19	
20	<pre># Mask out all other colors</pre>
21	<pre>mask = cv2.inRange(hsv, lower_color, upper_color)</pre>
22	
23	<pre># Multiply mask (0 values) with image</pre>
24	<pre>result = cv2.bitwise_and(frame, frame, mask = mask)</pre>

Color Filtering

By SharkDderivative work: SharkD [CC BY-SA 3.0 or GFDL], via Wikimedia Commons

Example: Color Filtering

Raw Data

Mask

Output

Raw Data

What are some limitations of this approach?

Question

Mask

Output

Image Processing Techniques

Basic Image Operations

- Thresholding
- Color Filtering
- Blurring
- Smoothing
- Background subtraction
- Edge Detection
- Corner Detection
- Feature Matching
- Haar Cascade Object Detection

•

Background Subtraction

Idea

Remove background from current image

Technical Implementation

- 1) Estimate background for time t
- 2) Subtract estimated background from current frame
- 3) Apply threshold to absolute difference

Background Model

This technique requires a background model that contains the static part of the scene. Best suited for a static camera.

Code

5	<pre>fgbg = cv2.createBackgroundSubtractorM0G2()</pre>
6	
7	while(cap.isOpened()):
8	ret, frame = cap.read()
9	
10	# Get the mask
11	fgmask = fgbg.apply(frame)
12	
13	<pre># Multiply mask (0 values) with image</pre>
14	<pre>result = cv2.bitwise_and(frame, frame, mask = fgmask)</pre>

OpenCV Docs: https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html

Example: Background Subtraction

Raw Data

<pre>fgbg = cv2.create</pre>	5
	6
while(cap.isOpened	7
→ ret, frame =	8
	9
# Get the mas	10
fgmask = fgbg	11
	12
<pre># Multiply mag</pre>	13
result = cv2.	14

Mask

Output

Raw Data

What are some limitations of this approach? (Other than requiring a more or less static camera)

Question

Mask

Output

Image Processing Techniques

- Thresholding
- Color Filtering
- Blurring
- Smoothing
- Background subtraction
- Edge Detection
- Corner Detection
- Feature Matching
- Haar Cascade Object Detection
- •

Convolution

Definition: Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel

Convolution

Definition: Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel

Convolution

Definition: Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel

Idea

Remove high frequency content (e.g. noise, edges, etc)

Technical Implementation

Convolve image with a normalized box filter i.e. take an average of all pixel under the kernel area and replace the central element with this average.

Kernel

Code

12	# Blur image using averaging filter kernel
13	<pre>blur1 = cv2.blur(frame,(3,3))</pre>
14	<pre>blur2 = cv2.blur(frame,(25,25))</pre>

Blurring

Operation	Kernel ω	Image result g(x,)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Kernel: https://en.wikipedia.org/wiki/Kernel_(image_processing)

Example: Blurring

Raw Data



12	# Blur ima
13	blur1 = cv
14	blur2 = cv

3x3 Kernel

25x25 Kernel

Question

Raw Data



Why would we want to do this?

3x3 Kernel

25x25 Kernel

Image Processing Techniques

- Thresholding
- Color Filtering
- Blurring
- Smoothing
- Background subtraction
- Edge Detection
- Corner Detection
- Feature Matching
- Haar Cascade Object Detection
- •

Idea

Determine the horizontal and vertical gradient, large gradient == edge

Technical Key

- 1) Apply gaussian filter to smooth the image and remove noise
- Find the gradients of the image using Sobel operator 2)
- Apply non max suppression to thin edges 3)
- Apply double threshold to determine strong and weak edges 4)
- Track edges to remove edges that are not connected to a strong edge 5)

Finding Gradients (Sobel Operator)

$$L_x = egin{bmatrix} +1 & 0 & -1 \ +2 & 0 & -2 \ +1 & 0 & -1 \end{bmatrix} L ext{ and } L_y = egin{bmatrix} +1 & +2 & +1 \ 0 & 0 & 0 \ -1 & -2 & -1 \end{bmatrix} L.$$

Code

12	# Find the edges
13	edges = cv2.Canny(frame, 100, 200)
14	
15	# Find the edges
16	<pre>blur = cv2.blur(frame,(5,5))</pre>
17	edges_blur = cv2.Canny(blur, 100, 200)



Idea

Determine the horizontal and vertical gradient, large gradient == edge

Technical Key

- 1) Apply gaussian filter to smooth the image and remove noise -
- Find the gradients of the image using Sobel operator 2)
- Apply non max suppression to thin edges 3)
- Apply double threshold to determine strong and weak edges 4)
- Track edges to remove edges that are not connected to a strong edge 5)

Finding Gradients (Sobel Operator)

$$L_x = egin{bmatrix} +1 & 0 & -1 \ +2 & 0 & -2 \ +1 & 0 & -1 \end{bmatrix} L ext{ and } L_y = egin{bmatrix} +1 & +2 & +1 \ 0 & 0 & 0 \ -1 & -2 & -1 \end{bmatrix} L.$$

Code

12	# Find the edges
13	edges = cv2.Canny(frame, 100, 200)
14	
15	# Find the edges
16	<pre>blur = cv2.blur(frame,(5,5))</pre>
17	edges_blur = cv2.Canny(blur, 100, 200)

Filter





Idea

Determine the horizontal and vertical gradient, large gradient == edge

Technical Key

- Apply gaussian filter to smooth the image and remove noise 1)
- Find the gradients of the image using Sobel operator -2)
- Apply non max suppression to thin edges 3)
- Apply double threshold to determine strong and weak edges 4)
- Track edges to remove edges that are not connected to a strong edge 5)

Finding Gradients (Sobel Operator)

$$L_x = egin{bmatrix} +1 & 0 & -1 \ +2 & 0 & -2 \ +1 & 0 & -1 \end{bmatrix} L ext{ and } L_y = egin{bmatrix} +1 & +2 & +1 \ 0 & 0 & 0 \ -1 & -2 & -1 \end{bmatrix} L.$$

Code

12	# Find the edges
13	edges = cv2.Canny(frame, 100, 200)
14	
15	# Find the edges
16	blur = cv2.blur(frame,(5,5))
17	edges_blur = cv2.Canny(blur, 100, 200)

Gradient Magnitude





Idea

Determine the horizontal and vertical gradient, large gradient == edge

Technical Key

- 1) Apply gaussian filter to smooth the image and remove noise
- 2) Find the gradients of the image using Sobel operator
- 3) Apply non max suppression to thin edges -
- 4) Apply double threshold to determine strong and weak edges
- 5) Track edges to remove edges that are not connected to a strong edge

Finding Gradients (Sobel Operator)

$$L_x = egin{bmatrix} +1 & 0 & -1 \ +2 & 0 & -2 \ +1 & 0 & -1 \end{bmatrix} L ext{ and } L_y = egin{bmatrix} +1 & +2 & +1 \ 0 & 0 & 0 \ -1 & -2 & -1 \end{bmatrix} L.$$

Code

12	# Find the edges
13	edges = cv2.Canny(frame, 100, 200)
14	
15	# Find the edges
16	blur = cv2.blur(frame,(5,5))
17	edges_blur = cv2.Canny(blur, 100, 200)

Non Max Suppression





Idea

Determine the horizontal and vertical gradient, large gradient == edge

Technical Key

- 1) Apply gaussian filter to smooth the image and remove noise
- 2) Find the gradients of the image using Sobel operator
- 3) Apply non max suppression to thin edges
- 4) Apply double threshold to determine strong and weak edges -
- 5) Track edges to remove edges that are not connected to a strong edge

Finding Gradients (Sobel Operator)

$$L_x = egin{bmatrix} +1 & 0 & -1 \ +2 & 0 & -2 \ +1 & 0 & -1 \end{bmatrix} L ext{ and } L_y = egin{bmatrix} +1 & +2 & +1 \ 0 & 0 & 0 \ -1 & -2 & -1 \end{bmatrix} L.$$

Code

12	# Find the edges
13	edges = cv2.Canny(frame, 100, 200)
14	
15	# Find the edges
16	<pre>blur = cv2.blur(frame,(5,5))</pre>
17	edges_blur = cv2.Canny(blur, 100, 200)

Double Thresholding



Idea

Determine the horizontal and vertical gradient, large gradient == edge

Technical Key

- 1) Apply gaussian filter to smooth the image and remove noise
- 2) Find the gradients of the image using Sobel operator
- 3) Apply non max suppression to thin edges
- 4) Apply double threshold to determine strong and weak edges
- 5) Track edges to remove edges that are not connected to a strong edge -

Finding Gradients (Sobel Operator)

$$L_x = egin{bmatrix} +1 & 0 & -1 \ +2 & 0 & -2 \ +1 & 0 & -1 \end{bmatrix} L ext{ and } L_y = egin{bmatrix} +1 & +2 & +1 \ 0 & 0 & 0 \ -1 & -2 & -1 \end{bmatrix} L.$$

Code

12	# Find the edges
13	edges = cv2.Canny(frame, 100, 200)
14	
15	# Find the edges
16	<pre>blur = cv2.blur(frame,(5,5))</pre>
17	edges_blur = cv2.Canny(blur, 100, 200)

Edge Tracking





Idea

Determine the horizontal and vertical gradient, large gradient == edge

Technical Key

- 1) Apply gaussian filter to smooth the image and remove noise
- 2) Find the gradients of the image using Sobel operator
- 3) Apply non max suppression to thin edges
- 4) Apply double threshold to determine strong and weak edges
- 5) Track edges to remove edges that are not connected to a strong edge

Finding Gradients (Sobel Operator)

$$L_x = egin{bmatrix} +1 & 0 & -1 \ +2 & 0 & -2 \ +1 & 0 & -1 \end{bmatrix} L ext{ and } L_y = egin{bmatrix} +1 & +2 & +1 \ 0 & 0 & 0 \ -1 & -2 & -1 \end{bmatrix} L.$$

Code

12	# Find the edges
13	edges = cv2.Canny(frame, 100, 200)
14	
15	# Find the edges
16	<pre>blur = cv2.blur(frame,(5,5))</pre>
17	edges_blur = cv2.Canny(blur, 100, 200)





Example: Edge Detection

Raw Data



12	# Find th
13	edges = c
14	
15	# Find th
16	blur = cv
17	edges_blu

Edge Detection

5x5 Blur -> Edge Detection

Example: Edge Detection

Raw Data



What are some of the limitations of this?

Edge Detection

5x5 Blur -> Edge Detection

Perception estimates the state of the environment

Image Processing

An image is processed through parameterized transformations.

Key: We define this function

Machine Learning

Gather large amounts of data a to learn or approximate the desired function.

Key: We learn this function

te

Perception estimates the state of the environment

Image Processing

An image is processed through parameterized transformations.

Key: We define this function

What are the pros and cons of image processing?

Machine Learning

Gather large amounts of data a to learn or approximate the desired function.

Key: We learn this function

te

Perception estimates the state of the environment

Image Processing Algorithms

An image is processed through parameterized transformations.

Key: We define this function

Pros: Does not require datasets at all Are easier to interpret by humans Most do not require heavy computation resources Libraries available to perform most standard functions

Cons: Encode relatively simple functions **Machine Learning**

Gather large amounts of data a to learn or approximate the desired function.

Key: We learn this function

te











Machine Learning

What happens if we don't know exactly how to define the function?







Regression

What is the relationship between a dependent and one or more independent variables?

Thrust

Regression



Thrust

What is the relationship between a dependent and one or more independent variables?

Machine Learning













Input

What happens if we have a much more complicated task?



Machine learning learns this function given enough data

3779

output = f(input)

Output

Plane, Car, Bird, Cat







How do we learn a function?



Neural Networks

Neural Networks - Training









Input







...

Label: Cat

Label: Dog



Output

Label: Dog

Label: Cat

Label: Cat

Label: Cat

. . .



Input



Neural Networks



Neural Networks - Data Augmentation



Original



Rotate



Translate





Neural networks needs LOTS of data

14 million labeled images

Data augmentation can increase the amount of data by adding slightly modified copies of already existing data.

Blur

Add Noise



Change **Brightness**

Zoom





60



Input

So how does this work?



Input

Neural Networks





Input Layer Hidden Layer





Input

Neural Networks

Output Layer



Output of a Neuron:

$$y = \sigma(w^T x + b)$$

y =output

- $\sigma =$ activation function
- w =weights
- $x = ext{input}$
- b = bias

Activation Function (ReLu)

$$\sigma(z) = egin{cases} 0 & z < 0 \ z & z \ge 0 \end{cases}$$







Neuron







Neural Networks - Feedforward



Neural Networks - Feedforward



Neural Networks - Visualization



Source: https://www.cs.ryerson.ca/~aharley/vis/fc/

0123456789



66

Neural Networks - Structure



Number of Layers

Activation functions



Source: Shruti Jadon

Learning Parameters:

- Learning rate
- Optimizer
- Batch Size
- Early stopping \bullet
- Number training epochs



Neural Networks - Weights

How do we compute these weights?





Neural Networks - Updating Weights

Computing networks weights:

1) For each observation in training set:

- Feedforward the observation \checkmark 2)
- 3) **Compute error**
- 4) **Run gradient descent to update weights**





Input Data

 $\mathbf{y}' = [0,1]$

Output Label



Neural Networks - Prediction Error

Computing networks weights:

- 1) For each observation in training set:
- Feedforward the observation \checkmark 2)
- 3) Compute error?
- 4) Run gradient descent to update weights

Prediction Error:



Input Data

 $\mathbf{y}' = [0,1]$

Output Label

Mean squared error:

$$error = \sum rac{1}{2} (\mathbf{y}' - \mathbf{y})^2$$

Also known as the cost function

 $x_1 = 1$



Neural Networks - Prediction Error

Computing networks weights:

- 1) For each observation in training set:
- 2) Feedforward the observation \checkmark
- 3) Compute error?
- 4) Run gradient descent to update weights

Prediction Error:



Input Data

 $\mathbf{y}' = [0,1]$

Output Label

$$error = \sum rac{1}{2} (\mathbf{y}' - \mathbf{y})^2 \ error = rac{1}{2} (y_1' - y_1)^2 + rac{1}{2} (y_2' - y_2)^2 \ error = rac{1}{2} (0 - 17.25)^2 + rac{1}{2} (1 - 53)^2 \ error = 1500.7813$$

 $x_1 = 1$

x_2	_
<i>w</i> 2	



Neural Networks - Gradient Descent

Computing networks weights:

- 1) For each observation in training set:
- Feedforward the observation 2)
- 3) Compute error 🧹
- Run gradient descent to update weights? 4)

Gradient descent:

$$error = \sum rac{1}{2} (\mathbf{y}' - \mathbf{y})^2$$

Function of the weights

To minimize the error, we can change the weights

$$w_k = w_k - \eta \left(rac{\partial error}{\partial w_k}
ight)$$




Neural Networks - Gradient Descent

Computing networks weights:

1) For each observation in training set:

- 2) Feedforward the observation \checkmark
- 3) Compute error
- 4) Run gradient descent to update weights ?

error = 1500.7813

Gradient descent:

Goal: Update the weights

$$error = \sum rac{1}{2} (\mathbf{y}' - \mathbf{y})^2 \ w_k = w_k - \eta \left(rac{\partial error}{\partial w_k}
ight)$$

Chain Rule

$$rac{\partial error}{\partial w_k} = rac{\partial error}{\partial y_1} imes rac{\partial y_1}{\partial z_1^{[2]}} imes rac{\partial z_1^{[2]}}{\partial w_1^{[2]}} + rac{\partial z_1^{[2]}}{\partial w_1^{[2]}}$$

 $x_1 = 1$

$x_2 =$



73

Neural Networks - Gradient Descent



x_1	=
-	

|--|



Neural Networks - Gradient Descent

Computing networks weights:

1) For each observation in training set:

- Feedforward the observation 2)
- 3) Compute error 🧹
- **Run gradient descent to update weights** 4)

error = 1500.7813

Gradient descent:

Goal: Update the weights

$$w_k = w_k - \eta \left(rac{\partial error}{\partial w_k}
ight)$$

$$egin{aligned} rac{\partial error}{\partial w_k} = 17.25 imes 1 imes 9 = 155.25 \ \eta = ext{learning rate} = 0.001 \end{aligned}$$

$$w_k = 0.25 - 0.001 \, (155.25) = 0.0948$$

 $x_1 = 1$



75







Classification



Yolov3: https://pjreddie.com/darknet/yolo/







MiconNet: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8481688

Neural Networks

Object Detection

Source: https://www.nvidia.com/en-us/self-driving-cars/drive-videos

Image Segmentation



NVIDIA Redtail: <u>https://github.com/NVIDIA-AI-IOT/redtail</u>

Openpose: https://github.com/CMU-Perceptual-Computing-Lab/openpose



S3D-UNet: https://link.springer.com/chapter/10.1007/978-3-030-11726-9_32



Perception Algorithms

Perception estimates the state of the environment

Image Processing Algorithms

An image is processed through transformations, filters, or algorithms. We can then use this information to infer something about that image. Key Difference: We define this function

Pros: Does not require huge labeled datasets Are easier to interpret by humans **Does not require heavy computation resources**

Cons: **Encode relatively simple functions**

Machine Learning

Gather large amounts of data and use this data to learn or approximate the desired function. We can then use this information to infer something about that image. Key Difference: We learn this function

Pros: Improves with more data Can learn complicated functions Can be used as an end-to-end an solution

Cons: Requires huge labeled datasets Requires heavy computation resources to train Difficult to interpret what they have learned Not robust to scenarios outside its training data

Research



Cost of Failure

