

Cats'heart, Cats'lake

— Xinyue Yu
121025

Sound is a great medium of communication between humans and animals. I have often seen artworks that achieve communication with insects through sonic transformation before. Compared to insects, various mammals, because of their larger brain capacity and greater domestication, can even understand human language directly, such as cats. So I was curious: how do cats react and feel when they hear their names called by humans? For this reason I carried out the following experiment: [link: https://youtube.com/playlist?list=PLsN6eU2xTDDzqQrCYrXqyJ0y0MQ4ruDsx](https://youtube.com/playlist?list=PLsN6eU2xTDDzqQrCYrXqyJ0y0MQ4ruDsx)

Experiment 1: Which one do cats prefer to, other interesting sounds or his owner's call?
Experimental Results:

- Cats loved being called by its owner's name, it would approach its owner and was happy.

Experiment 2: When the owner call the cat's name over the phone or a recording, could the cat tell the difference from call in person?

Experimental results:

- The cat could clearly recognize the difference between the sound made by a real person and a recording. Its reaction to the human and to the recording was completely different.

- Cats are very sensitive. It can also detect when its owner is talking extremely quietly on the phone behind the door and the telephone speaker in the room plays its owner's voice loudly and goes to the other side of the door to find its owner.

[link: https://www.youtube.com/watch?v=XR-km3RJ8m0](https://www.youtube.com/watch?v=XR-km3RJ8m0)

Experiment 3: A person sits motionless while he plays a recording calling out the cat's name, while another person sits opposite and calls out directly to the cat. What will the cat choose at this point?

Experimental results:

- The cat chose the owner who called its name, rather than the owner who played the recording. This shows that cats can indeed hear the difference between a recording and a real person.

[link: https://www.youtube.com/watch?v=6eHpt_fXIh4](https://www.youtube.com/watch?v=6eHpt_fXIh4)

The above experiment shows that cats are very dependent on humans and want them to call their names and keep them company. When cats hear their names called, they are very happy and will actively get close to humans. At the same time, cats are very intelligent and can clearly distinguish the difference between a human voice and a recording. And only the call of the owner himself and his company will satisfy them; recordings will either only make them more anxious to see their owner or they will simply be ignored. In addition to this, the cat had the following interesting behaviour: when the owner was sitting over there but called his name on the recording, he would sometimes crouch in front of the recording and wait for him instead of rubbing against him, while when only the recording was in front of him, he would mostly ignore it

and roam around. In all the experiments, he was only very active when his owner called his name directly and he rubbed against him. The dependence of cats on people is expressed entirely in their movement.

In this semester I have made an interactive installation that uses water waves to reflect the mood of a cat that is dependent on its owner. The mood cannot be seen or grasped, and the cat cannot speak or express itself. This installation can use water waves ripples to show the cat's heart, allowing the audience to watch the water waves fluctuate and indirectly experience the cat's heart of dependence on its owner. The audience can also interact with the installation, for example when they shout at it, touch it, or play a sound, the ripples will change in response.

[link: https://youtube.com/playlist?list=PLsN6eU2xTDDyOJmOfZkCPHmQXFnh7E-9b](https://youtube.com/playlist?list=PLsN6eU2xTDDyOJmOfZkCPHmQXFnh7E-9b)

In China, we often use water as a metaphor for moods: to describe a joyful and excited mood, we say "heart surging"——心潮澎湃; to describe a calm mood, we say "heart as calm as water"——心如止水; to describe an unmoved mood, we say "unperturbed by the waves"——波澜不惊; When describing melancholy, we say "falling into a whirlpool of melancholy"——掉入忧郁漩涡, etc. The fluctuation of mood is indeed very similar to water waves, and people can feel emotions through the fluctuation of water.

Procedure

First, I took video record of the reactions of my 3 cats in 3 separate scenarios.

1. The owner calls out and strokes
2. The owner sits still, only the shout recording is played
3. The owner leaves and only the shout recording is played

I input the filmed video into the installation. When the cat walks around, a large wave is created on the lake; when the cat stands motionless and watches somewhere, a whirlpool appears there; when the cat leaves, the lake is motionless again. I recorded the final effect of the installation corresponding to each cat in each scene.

I used touchdesigner to create the lake background and the ripple effect, and added the live tracking feature. I created the lake background by combining the ramp, geo and phong functions and adding the right amount of noise. The optical flow and wave ripples were then combined to create a real time tracking ripple effect. Wave ripples are generated based on the real-time tracking coordinates of the cat and the intensity of the movement. As the cat moves around the observation area, the water surface generates waves in response to its movements. As the trajectory and activity of the cat in different scenes is completely different, the waves generated by the installation are also completely different. (Ill 1. Screenshot of software playground)

)

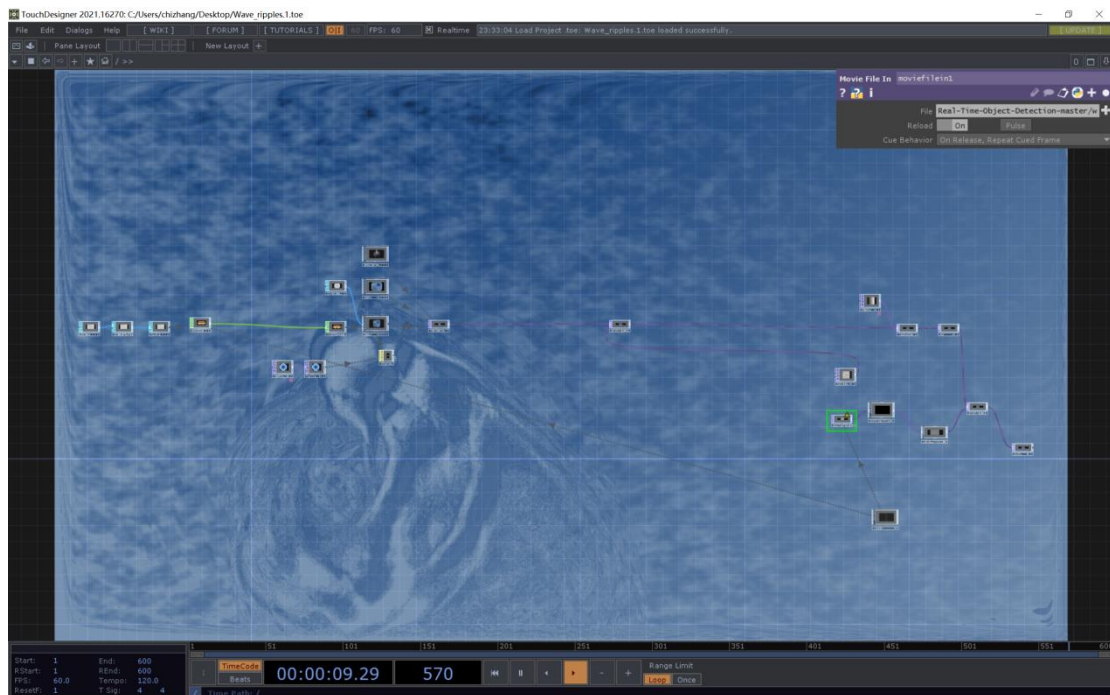
In addition to the real-time kinetic effects, I calculated the areas where the cat was most active during the experiment and drew a kernel density estimate of the

coordinate points of the active areas, using shades of blue to indicate the frequency of activity: where it stayed the most, it was the deepest blue; where it strolled past, it was a lighter blue. (Ill 2. Kernel density estimation maps for all scenarios and all cats)
Kernel density estimation maps are also used for reference in the device's interaction with humans, allowing people to visualise the cat's feelings of concern for them.

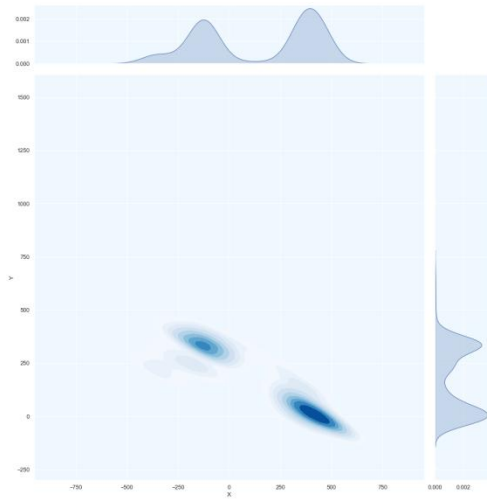
The principle of kernel density estimation map analysis: I used a python real-time tracking program I wrote to record their activity trajectory and the time spent at each point. [link: https://www.youtube.com/watch?v=6ZADfrQLGE&t=1s](https://www.youtube.com/watch?v=6ZADfrQLGE&t=1s)

After obtaining these coordinates, I aggregated the time and coordinate information of each segment and used the seaborn library as well as the matplotlib library to create a kernel density estimation map, where the origin of the coordinate system is the location where the owner is located. In this graph, the darker the colour, the more frequent the activity; the closer to the origin, the closer to the owner. (Ill 3. Screenshot of programm for object tracking)

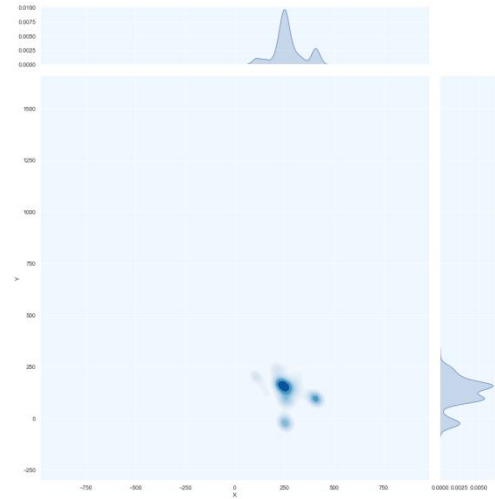
Attachments



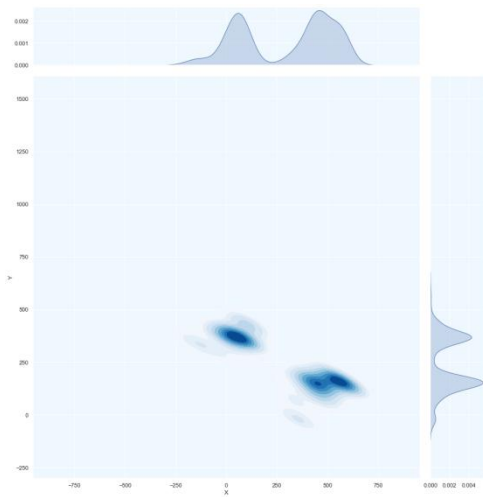
Ill 1. Screenshot of software playground



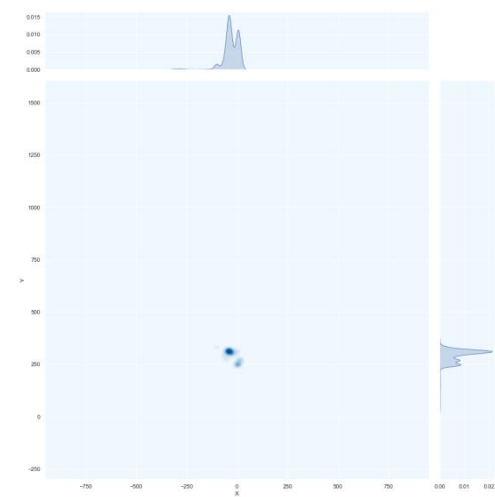
1. Scenario 1, cat Baobao



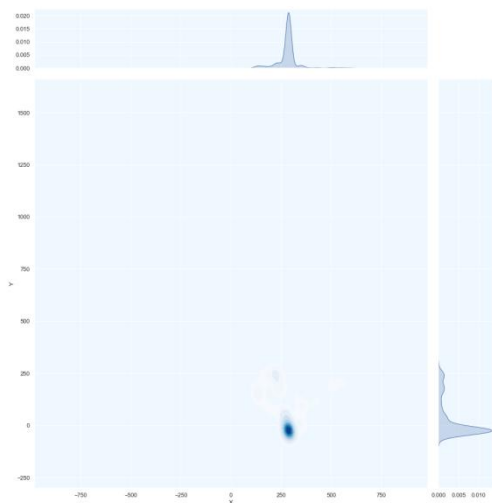
2. Scenario 1, cat Bingbing



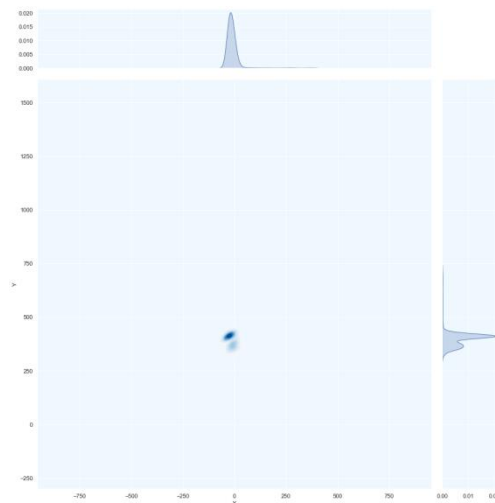
3. Scenario 1, cat Wulongcha



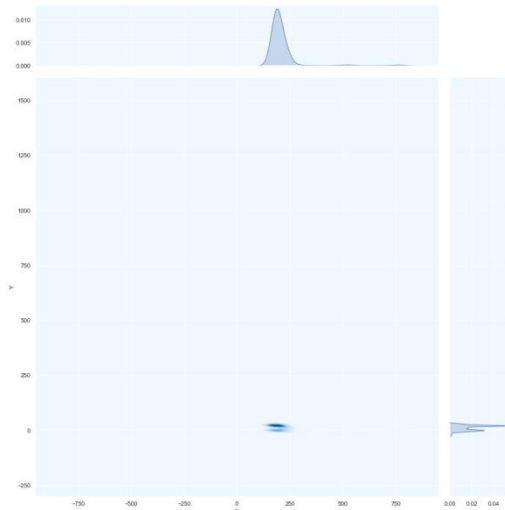
4. Senario 2, cat Baobao



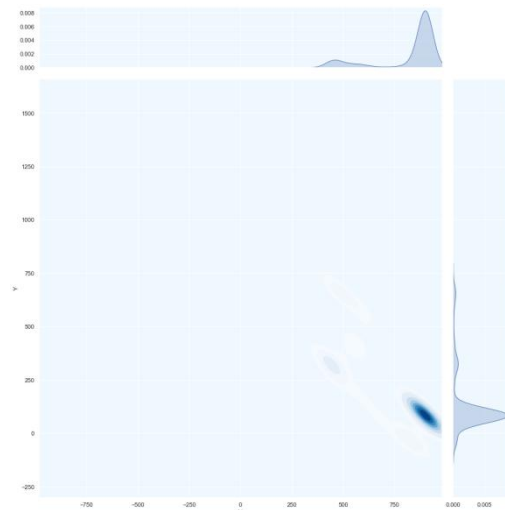
5. Senario 2, cat bingbing



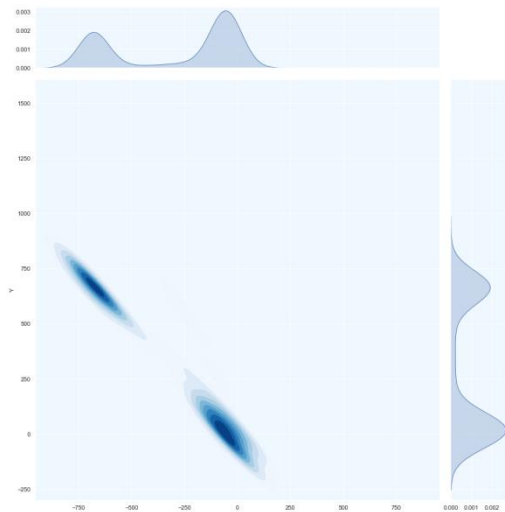
6. Senario 2, cat Wulongcha



7. Senario 3, cat Baobao



8. Senario 3, cat Bingbing

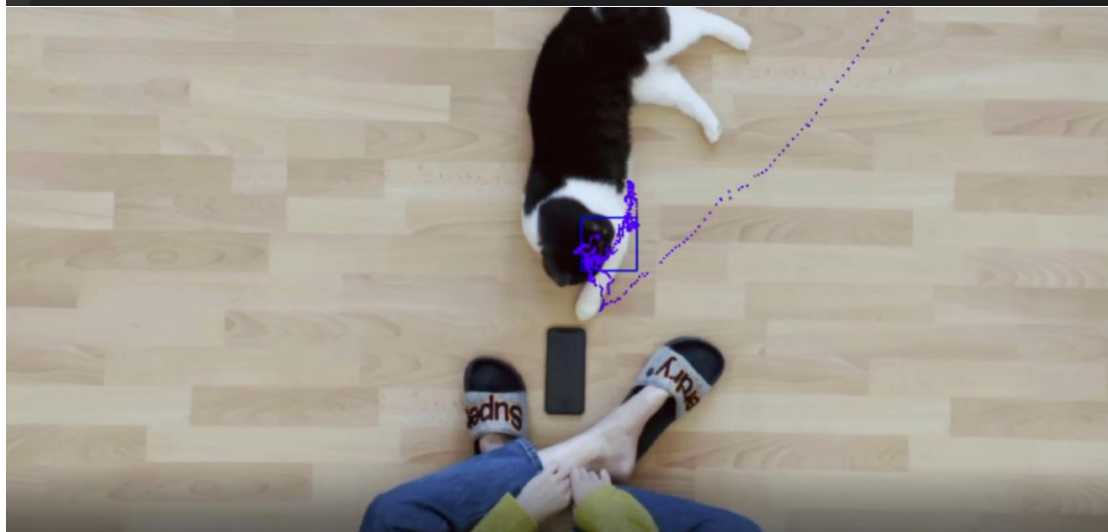


9. Senario 3, cat Wulongcha

Ill 2. Kernel density estimation maps for all scenarios and all cats

```
EXPLORER
REAL-TIME-OBJECT-DETECTION...
detect_by_drawing.py
13 #tracker = cv2.TrackerMOSSE_create()
14
15 #####
16
17 path = 'wuolongcha/2/'
18 cap = cv2.VideoCapture('./'+path+'source.mp4')
19 # TRACKER INITIALIZATION
20 success, frame = cap.read()
21 cv2.namedWindow("Tracking", cv2.WINDOW_NORMAL)
22 bbox = cv2.selectROI("Tracking", frame, False)
23 #import pdb; pdb.set_trace()
24 tracker.init(frame, bbox)
25 point_list = []
26
27
28 def drawBox(img, bbox, point_list):
29     x, y, w, h = int(bbox[0]), int(bbox[1]), int(bbox[2]), int(bbox[3])
30     point_x, point_y = int(x+w/2), int(y+h/2)
31     point_list.append((point_x, point_y))
32     cv2.rectangle(img, (x, y), ((x + w), (y + h)), (255, 0, 0), 2, 2)
33     # cv2.putText(img, "Tracking", (100, 75), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
34     return point_list
35
36
37 while True:
38
39     timer = cv2.getTickCount()
40     success, img = cap.read()
41     success, bbox = tracker.update(img)
42
43     if success:
44         point_list = drawBox(img, bbox, point_list)
45         for i in point_list:
46             cv2.circle(img, (i[0], i[1]), radius=2, color=(255, 0, 80), thickness=-2)
47
48
49 # cv2.rectangle(img, (15, 15), (280, 90), (255, 0, 255), 2)
50
51 # cv2.putText(img, "FPS:", (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 255), 2);
52 # cv2.putText(img, "Status:", (20, 75), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 255), 2);
53
54
55 fps = cv2.getTickFrequency() / (cv2.getTickCount() - timer)
56 if fps>60: myColor = (20, 230, 20)
57 elif fps>20: myColor = (230, 20, 20)
58 else: myColor = (20, 20, 230)
59
60
61 cv2.imshow("Tracking", img)
62 if cv2.waitKey(1) & amp; amp; ord('q'):
63     fields = ['X', 'Y']
64     with open(path, 'w') as f:
65         # using csv.writer method from CSV package
66         write = csv.writer(f)
67         write.writerow(fields)
68         write.writerow(point_list)
69         break
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
EXPLORER
REAL-TIME-OBJECT-DETECTION...
detect_by_drawing.py
37 while True:
38
39     timer = cv2.getTickCount()
40     success, img = cap.read()
41     success, bbox = tracker.update(img)
42
43     if success:
44         point_list = drawBox(img, bbox, point_list)
45         for i in point_list:
46             cv2.circle(img, (i[0], i[1]), radius=2, color=(255, 0, 80), thickness=-2)
47
48
49 # cv2.rectangle(img, (15, 15), (280, 90), (255, 0, 255), 2)
50 # cv2.putText(img, "FPS:", (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 255), 2);
51 # cv2.putText(img, "Status:", (20, 75), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 255), 2);
52
53
54
55 fps = cv2.getTickFrequency() / (cv2.getTickCount() - timer)
56 if fps>60: myColor = (20, 230, 20)
57 elif fps>20: myColor = (230, 20, 20)
58 else: myColor = (20, 20, 230)
59
60
61 cv2.imshow("Tracking", img)
62 if cv2.waitKey(1) & amp; amp; ord('q'):
63     fields = ['X', 'Y']
64     with open(path, 'w') as f:
65         # using csv.writer method from CSV package
66         write = csv.writer(f)
67         write.writerow(fields)
68         write.writerow(point_list)
69         break
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



III 3. Screenshot of programm for object tracking