

# AUTOMATION OF PET BEHAVIOR IMPROVEMENT AND SERVICE TRAINING USING ARTIFICIAL INTELLIGENCE AND COMPUTER VISION

Shengyu Wang<sup>1</sup>, Jonathan Sahagun<sup>2</sup>

<sup>1</sup>St. Margaret's Episcopal School, 31641 La Novia Ave, San Juan  
Capistrano, CA 92675

<sup>2</sup>Computer Science Department, California State Polytechnic University,  
Pomona, CA 91768

## **ABSTRACT**

*Owning a dog brings with it a multitude of responsibilities, chief among them being the essential task of dedicating ample time to training [1]. While the specific time commitment may vary, effective training necessitates steadfast dedication and effort. Unfortunately, many dog owners struggle to allocate sufficient time, leading to untrained canine behavior [2]. Research indicates that approximately 75% of dogs worldwide lack proper training. Effective training serves as the bedrock of a harmonious human-canine relationship, fostering mutual understanding and safety [3]. Despite its paramount importance, training demands patience, consistency, and perseverance, qualities often underestimated by dog owners. Addressing this issue requires a shift in owners' perspectives, underscoring the significance of prioritizing training through professional guidance and obedience classes. Ultimately, investing in training not only enhances the quality of life for dogs but also strengthens the bond between owners and their cherished companions, showcasing the dedication necessary to ensure canine well-being in an ever-changing world.*

## **KEYWORDS**

*Dog Training, Artificial Intelligence, Human-Canine Bond*

## **1. INTRODUCTION**

Owning a dog comes with a lot of responsibilities, and one of those responsibilities is to spend the appropriate amount of time training your dog. Now, the specific time can vary from person to person, but it without a doubt takes time and commitment. Sometimes people aren't able to commit that much time to training their dogs, which results in the dog's behavior being untrained. On average 75% of the world's dogs aren't properly trained [4].

The significance of training cannot be overstated. It forms the cornerstone of a harmonious relationship between a dog and its owner, fostering mutual understanding and respect. Through training, dogs learn vital skills such as obedience, socialization, and appropriate behavior in diverse environments. Moreover, a well-trained dog is safer, both for itself and for those around it.

However, training demands patience, consistency, and perseverance. It is a gradual process that requires regular practice and reinforcement. Many dog owners underestimate the time and effort required, leading to a staggering number of untrained dogs globally.

Addressing this issue necessitates a shift in perspective regarding the responsibilities of dog ownership. Owners must recognize the importance of training and prioritize it accordingly. Seeking professional guidance or enrolling in obedience classes can provide valuable support in this endeavor.

Ultimately, the commitment to training not only enhances the quality of life for dogs but also fosters stronger bonds between owners and their beloved companions. It is a testament to the dedication and care required to ensure the well-being of our canine companions in an ever-evolving world.

The proposed solution to address the challenge of insufficient time for dog training is the development of a revolutionary device: the AI Pup Trainer [8]. This innovative technology aims to alleviate the burden on dog owners by streamlining the training process with minimal effort required from them. The AI Pup Trainer serves as a reliable assistant, enabling dog owners to effectively train their pets even when they are unable to dedicate extensive hours solely to training.

What sets the AI Pup Trainer apart is its advanced capabilities in detecting and interpreting the actions of dogs. Through sophisticated algorithms and sensors, the device can analyze the behavior of the dog and provide tailored training feedback based on specific commands. This autonomous functionality empowers dog owners by automating aspects of the training process, thus mitigating the challenges posed by time constraints.

By leveraging artificial intelligence, the AI Pup Trainer offers a dynamic approach to dog training, adapting to the unique needs and behaviors of individual dogs. Its ability to provide real-time feedback and reinforcement enhances the learning experience for both dogs and owners, fostering a more efficient and rewarding training journey.

The AI Pup Trainer represents a paradigm shift in the realm of dog training, offering a solution that is not only effective but also accessible and user-friendly [9]. With this innovative device, dog owners can embark on a transformative journey towards nurturing well-trained and disciplined canine companions, enriching the bond between humans and their beloved pets.

The dog detection experiment has proven its reliability in detecting the presence of a dog with a high degree of confidence. Through rigorous testing and analysis, the experiment consistently demonstrates its ability to accurately identify dogs in various conditions and environments. Its dependable performance instills confidence in its outcomes, making it a valuable tool for tasks requiring precise dog detection.

The current dispensing mechanism operates with a fail rate of 20%, indicating that one out of every five attempts may encounter issues such as jams or malfunctions. However, this rate presents an opportunity for improvement through redesigning the mechanism. By implementing adjustments and enhancements, the dispensing system can mitigate the occurrence of jams significantly. Redesigning may involve optimizing the internal components, refining the feeding mechanism, or employing more durable materials to enhance reliability. These modifications aim to streamline the dispensing process, ensuring smoother operation and reducing the frequency of disruptions or errors.

## **2. CHALLENGES**

In order to build the project, a few challenges have been identified as follows.

### **2.1. The Image Detection**

The image detection aspect, pivotal yet prone to significant issues, might encounter challenges if it doesn't perform optimally. Utilizing YOLO V8 for image detection, I could enhance accuracy by incorporating additional images of both sitting and non-sitting dogs [10]. This strategy aims to refine the software's capability and improve overall detection precision.

### **2.2. The Servo Motor**

Another major component that could cause some problems is the servo motor not being able to dispense the treat correctly. This could be due to the python code on the raspberry pi not rotating the right amount to dispense the treat properly. It can be fixed just by inputting the correct amount into the python code to make the motor spin in a 180° angle.

### **2.3. Raspberry Pi Hardware**

Achieving seamless integration of Raspberry Pi hardware with our app poses a potential challenge [11]. Establishing effective communication channels between them entails exploring diverse options, including Bluetooth connectivity, configuring the Raspberry Pi as a local hotspot, or equipping it with cellular capabilities. Each approach brings forth distinct considerations, ranging from range and reliability to data usage. Optimal selection among these methods depends on various factors such as intended usage scenarios, user preferences, and technical limitations. The paramount objective is to ensure robust connectivity between the Raspberry Pi and our app, thereby delivering a user experience that is both seamless and dependable.

## **3. SOLUTION**

My program seamlessly integrates three key components: an Image detection model, a treat dispenser program, and a command-giving program [12]. The program flow operates as follows: Initially, the camera captures an image to detect the presence of a dog. If a dog is detected, it proceeds to execute the "sit" command. Conversely, if no dog is detected, the program waits for one minute before capturing another image. Following the "sit" command, another image is captured to verify if the dog is sitting. If the dog is sitting, treats are dispensed; otherwise, the program returns to the initial stage to detect the presence of a dog. Subsequently, after dispensing treats, the program waits for approximately 20 seconds before initiating the process anew, effectively restarting the cycle.

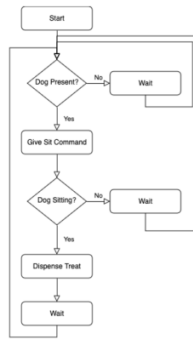


Figure 1. Overview of the solution

One component is the image detection model, its purpose is to detect whether or not there is a dog in the image. To implement this system I used the YOLO V8 model to specifically train it to detect whether there is a dog present or sitting [13].

```

52
53 def isDogPresent(image):
54     """
55     Checks if dog is sitting.
56     Returns true if dog is present, returns false otherwise. Also returns
57     the confidence number
58
59     Keyword arguments:
60     image -- an image from cv2
61
62     returns bool, float
63     """
64     results = model.predict(source=image)
65     dog_found = False
66     dog_conf = 0
67     for result in results:
68         for conf, cls in zip(result.boxes.conf, result.boxes.cls):
69             object_name = names[int(cls)]
70             object_conf = float(conf)
71
72             if int(cls) == DOG_CLASS_ID and object_conf >= CONFIDENCE_THRESHOLD:
73                 dog_found = True
74                 print('dog found with conf', conf)
75                 return dog_found, object_conf
76     return False, 0.0
77
78 def isDogSitting(image):
79     """
80     Checks if dog is sitting.
81     Returns true if dog is sitting, returns false otherwise.
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 # Program Settings
113 DOG_PRESENT_DELAY = 2 # In Seconds
114 DOG_SIT_DELAY = 4 # In Seconds
115 NEXT_TREAT_DELAY = 10 # In Seconds
116
117 CONFIDENCE_THRESHOLD = 0.5
118
119 DEBUG_VOBOSE = True
120
121 # Program Setup
122 ## Stepper Motor Setup
123 motorA = DigitalOutputDevice(25) # Green Wire, GPIO Pin 25
124 motorB = DigitalOutputDevice(8) # Blue Wire, GPIO Pin 8
125 motorC = DigitalOutputDevice(7) # Yellow Wire, GPIO Pin 7
126 motorD = DigitalOutputDevice(1) # White Wire, GPIO Pin 1
127
128 STEP = 0.703125 # DO NOT CHANGE
129 t = 0.0025
130
131 ## Tensorflow Lite Setup - Dog Sitting
132 interpreter = tf.lite.Interpreter(model_path="model_unquant.tflite")
133 interpreter.allocate_tensors()
134
135 input_details = interpreter.get_input_details()
136 output_details = interpreter.get_output_details()
137
138 labels = ['not_sitting', 'sitting']
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

```

Figure 2. Screenshot of code 1

```

## YoloV8 Model Setup - Dog Detected
model = YOLO('yolov8n.pt')
names = model.names
DOG_CLASS_ID = 16

## Audio Setup
sit_command_audio_path = "sit.mp3"
sit_command_audio_duration = 1
mixer.init()
mixer.music.load(sit_command_audio_path)

```

Figure 3. Screenshot of code 2

The YOLO V8 model is imported into the Raspberry Pi and then is called upon when the camera takes a picture to detect whether the dog is there or not. The variable model is created to import the Yolo v8 model correctly into the code. The variable DOG\_CLASS\_ID is made to identify the correct index in which the detection of the dog is. This is because the YOLO V8 model is also trained on identifying a lot of objects and putting them into a numbered list, specifying each object as an index, this makes it so that if you wanted to single out an object you would have to specify the specific index it is listed as. With the dog then being detected the model will wait until the program itself is repeated to run again.

The treat dispenser program is run by a servo motor that is programmed to rotate at specific intervals. The rotation of the motor is decided by my python program that will only run it on a conditional. This motor is attached to a 3D printed model of a dispenser to help it dispense the treat [14].

```

102
103 def dispenceTreat(angle = 180):
104     """
105     Turns motor in a 180 angle and drops treats onto floor
106     """
107     num_steps = int(angle / STEP)
108     for i in range(num_steps):
109         motorA.blink(t,t, False)
110         motorB.blink(t,t, False)
111         motorC.blink(t,t, False)
112         motorD.blink(t,t, False)
113
114
# Program Setup
## Stepper Motor Setup
motorA = DigitalOutputDevice(25) # Green Wire, GPIO Pin 25
motorB = DigitalOutputDevice(8) # Blue Wire, GPIO Pin 8
motorC = DigitalOutputDevice(7) # Yellow Wire, GPIO Pin 7
motorD = DigitalOutputDevice(1) # White Wire, GPIO Pin 1
STEP = 0.703125 # DO NOT CHANGE
t = 0.0025

```

Figure 4. Screenshot of code 3

```

while True:
    ret, image = camera.read()
    if not ret:
        print('no image')
        break
    cv2.imshow('camera feed', image)
    cv2.waitKey(100)
    delay_time = 5
    if time.time() - start_time > delay_time:
        if DEBUG_VERBOSE: print('Processing Image...')
        dog_present_conf = isDogPresent(image)
        if dog_present:
            if DEBUG_VERBOSE: print('Dog is present')
            giveSitCommand()
            if DEBUG_VERBOSE: print('Sit command is given')
            dog_sitting = isDogSitting(image)
            if dog_sitting:
                if DEBUG_VERBOSE: print('Dog is sitting')
                dispenceTreat()
                time.sleep(1)
                if DEBUG_VERBOSE: print('Treat Dispenced')
                wait(NEXT_TREAT_DELAY)
            else:
                wait(DOG_SIT_DELAY)
        else:

```

Figure 5. Screenshot of code 4

In the first image it shows the function that calls the servo motor to rotate in a 180-degree angle in order to dispense the treat. In the second image below, it shows the pins that each stepper motor wire is connected to.

The program begins by defining a function called dispenceTreat. This function is responsible for dispensing treats using a motor. When called, the motor turns precisely 180 degrees, causing treats to drop onto the floor. The function calculates the number of motor steps required based on

the specified angle. Four GPIO pins (motorA, motorB, motorC, and motorD) control the motor's movement. The STEP constant is set to 0.703125 (a critical value for motor control). A delay of 5 seconds occurs before processing the next image.

In the program setup section, GPIO pins are assigned to each motor (e.g., motorA corresponds to GPIO pin 25). The t variable is set to 0.0025. The program enters an infinite loop (while True) to continuously process camera images.

During image processing, the code reads an image from the camera feed. If an image is successfully retrieved, it briefly displays it (cv2.imshow) and then proceeds to analyze it. The isDogPresent function (not shown here) evaluates whether a dog is in the frame and the isDogSitting function determines if the dog is sitting. If a dog is detected and meets the specified conditions, the dispenseTreat function is invoked.

## 4. EXPERIMENT

### 4.1. Experiment 1

To set up this experiment I can have a dog run across the camera as it is taking an image. This is to test the accuracy of its identification and make sure that no matter the frame the AI will be able to detect it. It is important that this works well as it is the basis of my design and without accurate detection the program won't run as well as it should be. The experiment is set up this way to determine if the detection model is good enough to detect a dog that is blurry in the image.

| Experiments  | Confidence | Correct |
|--------------|------------|---------|
| Experiment 1 | 90%        | Yes     |
| Experiment 2 | 88%        | Yes     |
| Experiment 3 | 89%        | No      |
| Experiment 4 | 91%        | No      |
| Experiment 5 | 78%        | Yes     |

Figure 6. Figure of experiment 1

While high confidence scores may instill trust, they don't always guarantee precision, as evidenced in Experiments 3 and 4. Conversely, Experiment 5, despite lower confidence levels, managed to produce a correct outcome. These outcomes underscore the nuanced nature of model assessment.

In evaluating experimental data, it's imperative to weigh both confidence and correctness. A model's confidence score serves as a valuable indicator but must be contextualized within the broader framework of accuracy. Researchers and developers must navigate this trade-off carefully, understanding that optimal performance isn't solely dictated by confidence levels. By considering both dimensions, we gain deeper insights into the model's behavior and its implications for real-world applications.

### 4.2. Experiment 2

Another potential blind spot lies in the dispensing mechanism of the treat. Due to the treats funneling through a glass jar positioned at the top of the design, there is a risk of treats getting stuck in the dispenser, leading to machine breakage and jamming. To address this issue, I plan to simulate real-world conditions by setting up an image of a dog in front of the design and running

it through the program multiple times. This experiment aims to quantify the frequency of jamming incidents and assess the reliability of the device under operational conditions. By systematically testing the device's performance in a controlled setting, we can identify potential weaknesses in the dispensing mechanism and implement necessary improvements to enhance its durability and functionality.

| Experiments  | Jammed | Not Jammed |
|--------------|--------|------------|
| Experiment 1 | No     | Yes        |
| Experiment 2 | No     | Yes        |
| Experiment 3 | No     | Yes        |
| Experiment 4 | Yes    | No         |
| Experiment 5 | No     | Yes        |

Figure 7. Figure of experiment 2

The experiment findings reveal that the dispensing mechanism of the treat dispenser exhibits a failure rate of 20%. This failure rate signifies a notable proportion of instances where the dispenser does not effectively dispense treats as intended. Such occurrences pose significant implications for the reliability and functionality of the treat dispenser system.

Implementing a redesigned rotating mechanism that addresses these underlying issues can significantly reduce the failure rate and improve the overall functionality of the treat dispenser. This may involve optimizing gear ratios, enhancing motor torque, incorporating anti-jamming mechanisms, or redesigning component geometries to promote smoother rotation and reliable treat dispensing.

## 5. RELATED WORK

The article titled "Dog behavior classification with movement sensors placed on the harness and the collar" utilizes motion sensors positioned on the harness and collar to discern a dog's behavior [5]. In contrast, our approach employs a camera initially to detect if a dog is sitting, with the potential to extend our technology for identifying other dog behaviors in the future.

Similarly, in the article "Dog Sit! Domestic Dogs (Canis familiars) Follow a Robot's Sit Commands," the focus is on determining if dogs respond to commands from a robot and loudspeaker [6]. Conversely, our research aims to ascertain whether dogs can be trained using our method, thus testing its efficacy on untrained dogs.

Furthermore, in the article "Dog Behavior Recognition Based on Multimodal Data from a Camera and Wearable Device," the authors utilize Yolo to identify dog behaviors, resembling AI Pup Trainer [7]. However, our approach diverges as we focus on training dogs, while their methodology primarily monitors general dog behavior and health.

## 6. CONCLUSIONS

Transitioning from Raspberry Pi 3 to Raspberry Pi 5 presents a viable solution to address constraints in speed and accuracy associated with the dog detection model. The enhanced capabilities of Raspberry Pi 5 allow for accommodating heavier models, thereby facilitating higher accuracy in dog detection. This upgrade not only accelerates processing times but also

enables the deployment of more sophisticated algorithms, contributing to improved performance and precision in the detection process.

However, an ongoing challenge with Raspberry Pi devices lies in their connectivity to the internet for data uploads to the database and subsequent transmission to the app. Current procedures typically require users to connect the device to their home Wi-Fi using a monitor, mouse, and keyboard, lacking the user-friendliness crucial for seamless IoT device integration into home networks. Simplifying the connectivity process is imperative for enhancing user experience and promoting widespread adoption of IoT technology in various settings.

Despite these challenges, the AI Pup Trainer device and app exhibit high accuracy in detecting dogs and assessing their sitting behavior. While the training of dogs may require additional evaluation time, the AI Pup Trainer device functions effectively and merely needs to address the jamming issue, which can be easily overcome.

## REFERENCES

- [1] Hiby, E. F., N. J. Rooney, and J. W. S. Bradshaw. "Dog training methods: their use, effectiveness and interaction with behaviour and welfare." *Animal welfare* 13.1 (2004): 63-69.
- [2] Luescher, Andrew U. "Canine behavior and development." *Canine and Feline Behavior for Veterinary Technicians and Nurses* (2014): 30-50.
- [3] Maharaj, Nandini, Arminee Kazanjian, and C. J. Haney. "The human–canine bond: A sacred relationship." *Journal of spirituality in mental health* 18.1 (2016): 76-89.
- [4] Phenix, Annie. *The Midnight Dog Walkers: Positive Training and Practical Advice for Living With Reactive and Aggressive Dogs*. Fox Chapel Publishing, 2016.
- [5] Gidaris, Spyros, and Nikos Komodakis. "Locnet: Improving localization accuracy for object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- [6] Kumpulainen, Pekka, et al. "Dog behaviour classification with movement sensors placed on the harness and the collar." *Applied Animal Behaviour Science* 241 (2021): 105393.
- [7] Kim, Jinah, and Namme Moon. "Dog behavior recognition based on multimodal data from a camera and wearable device." *Applied sciences* 12.6 (2022): 3199.
- [8] Allen, Greg. "Understanding AI technology." *Joint Artificial Intelligence Center (JAIC) The Pentagon United States* 2.1 (2020): 24-32.
- [9] Todd, Zazie. "Barriers to the adoption of humane dog training methods." *Journal of Veterinary Behavior* 25 (2018): 28-34.
- [10] Hussain, Muhammad. "YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection." *Machines* 11.7 (2023): 677.
- [11] Maksimović, Mirjana, et al. "Raspberry Pi as Internet of things hardware: performances and constraints." *design issues* 3.8 (2014): 1-6.
- [12] Srivastava, Shrey, et al. "Comparative analysis of deep learning image detection algorithms." *Journal of Big data* 8.1 (2021): 66.
- [13] Alruwaili, Madallah, et al. "Deep Learning-Based YOLO Models for the Detection of People With Disabilities." *IEEE Access* (2023).
- [14] Rungrojwittayakul, Oraphan, et al. "Accuracy of 3D printed models created by two technologies of printers with different designs of model base." *Journal of Prosthodontics* 29.2 (2020): 124-128.
- [15] Balaji, Subramanian, Karan Nathani, and RathnasamySanthakumar. "IoT technology, applications and challenges: a contemporary survey." *Wireless personal communications* 108 (2019): 363-388.