令和元年度　特別研究報告書

超音波距離センサーを用いた
老人見守り

龍谷大学　理工学部　情報メディア学科

T170488　新造　一友

指導教員　三好　力　教授

内容梗概

近年、65 歳以上の一人暮らしや、老夫婦のみの世帯は年々増加傾向にあり、老々介護や孤独死などが大きな問題となってきている。特に、高齢者の一人暮らしは孤独死に繋がりかねないため、定期的に様子を見に行く人も多いだろう。しかし、遠方に住んでいる、近くても忙しくて定期的に見に行くこともできない人達もいる。その人たちのために遠くからでも安否確認を行える「高齢者見守りサービス」が様々な企業から提供されており、それらを当人に合った選び方をするのが大事になる。しかし、見守り方が最適であっても、室内カメラ等本人が納得しない場合もある。

本研究ではカメラほど忌避感を与えずに主に高齢者の出入りを見守る方法の一つとして、超音波距離センサーを用いた見守りの方法を提示し、実用に向くか検証した。

目次

第1章　研究背景

平成 29 年の国民生活基礎調査[1]では、18 歳未満の児童がいる世帯が核家族の世帯は 82.7%を占めており、家庭の核家族化が顕著になりつつある。そのため、高齢者だけで住んでいる世帯が増えてきている。

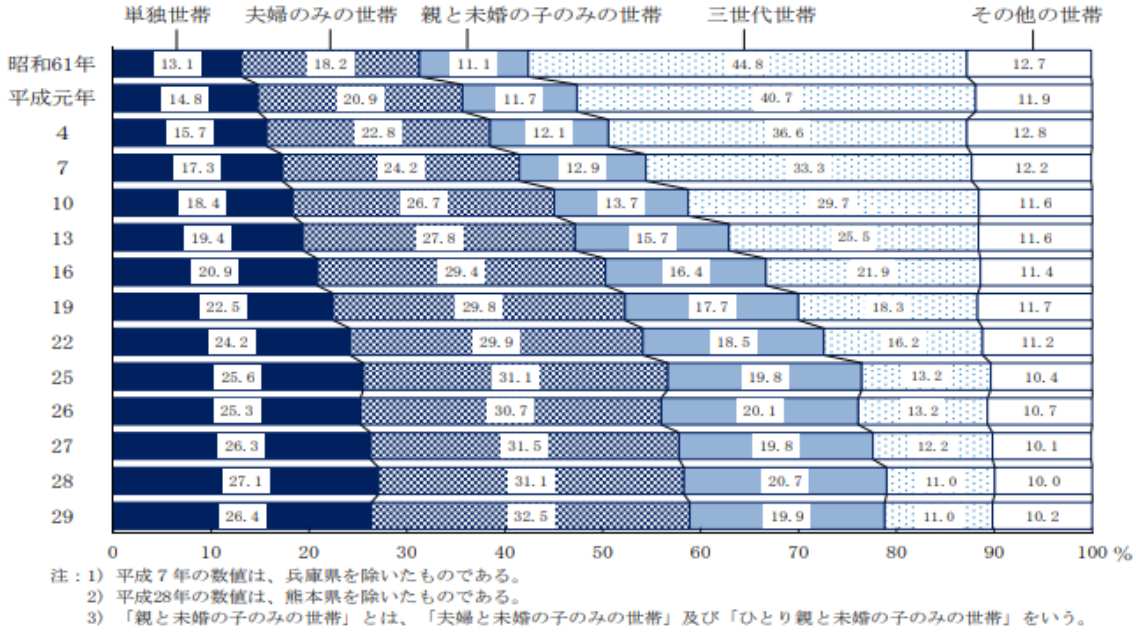

図１　[1]より 65 歳以上の者のいる世帯の世帯構造の年次推移

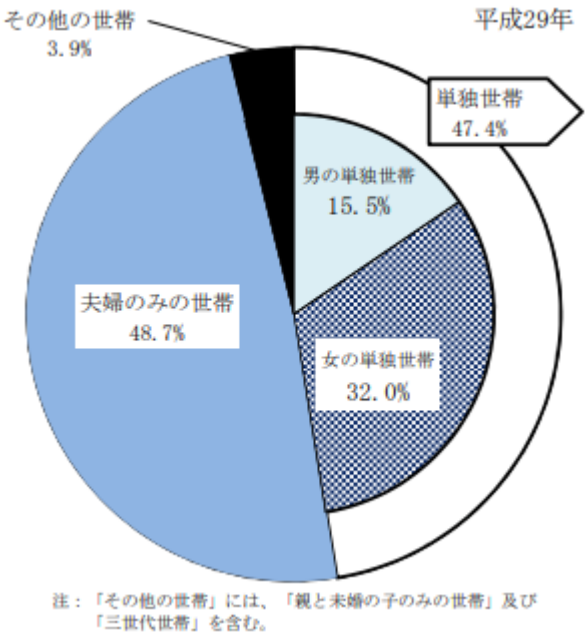

図２　[1]より高齢者世帯の世帯構造

上図より、65 歳以上の一人暮らしや、老夫婦のみの世帯は年々増加傾向にあり、老々介護や孤独死などが大きな問題となってきている。

特に、高齢者の一人暮らしは孤独死に繋がりかねないため、定期的に様子を見に行く人も多いだろう。しかし、遠方に住んでいたり、忙しくて定期的に見に行くこともできない人達もいる。そう遠くない位置に住んでいるが、近年のコロナウィルス感染症対策のため、世間体などを気にして以前ほど頻繁に様子を見に行けない家庭も少なからず存在するだろう。

その人たちのために遠くからでも安否確認を行える「高齢者見守りサービス」が様々な企業から提供されており、警備会社のスタッフが訪問して安否確認を行うなどの手厚いものからグッズ一つの手軽なものまでそろっている。見守る場所も機械によって様々で風呂やトイレ等の生理的側面から生活を、冷蔵庫やポットなどから食生活を、家の出入りから規則的な生活をしているかを見守ったりするものもある。これらを当人に合った選び方をするのが大事になる。

例えば高齢者の一人暮らしの場合は、日々の食事を近くのスーパーやコンビニに買いに行く例が十分にあり、出入りを見守るだけでも効果があるといえる。また、見守り方が最適であっても本人が納得しない場合もある。私が周囲からよく聞く見守り方法は室内カメラだが、見守られる高齢者の方が見られてる気がして落ち着かないなどと設置を渋る、無理に設置した場合外されることも多々ある。そこでカメラほど忌避感を与えずに主に高齢者の出入りを見守る方法の一つとして、超音波距離センサーを用いた見守りの方法を提案する。

第2章　既存技術

2.1　カメラを使わない見守りサービス

以下で紹介する見守りサービスは見守られる高齢者にとって違和感が少なく、見守る側にとっても複雑な操作が必要なく導入し易いサービスの中で、高齢者の出入りの見守りに適している一例である。

2.1.1　i ポット

象印マホービン株式会社が提供している見守りサービスである。電源をオンオフや給湯、再沸騰、使用等のタイミングで使用状況をサーバーへ送信し、1 日 2 回メールで使用状況を通知する。その他にもホームページで 1 週間分のグラフを確認することができる。また、お出かけボタンを押すことで、見守り側に外出を教え、使用してない事を不安に思わせないための機能も付いている。[2]

図 3　i ポット

2.1.2　開閉センサー

株式会社リンクジャパンが提供している見守りサービスの一つである。同社の eHub を設置しているとドアの開閉や、入って一定時間出てこないことを感知し、メールで通知を行う。[3]

図 4　開閉センサー

2.1.3　HelloLight

ハローライト株式会社が提供している見守りサービスである。電球に SIM を内蔵し、点灯感知や期間感知を行い、登録したアドレスにメール送信を行う。[4]

図 5　HelloLight

2.2 問題点

今回出入りを見守るに当たって問題となる部分がある。例えば 2.1.2 の開閉センサーを使用している場合、換気で開け放たれた状態やドアが閉まりきらずにセンサーが動かない状態で放置された時、開け放っている警告は入るだろうがその間の閉まるまでの情報が入ってこなくなる。2.1.3 の場合でも暗い時間帯の外出の際電気を消すか消さないかで精度が変わり、出入りを見守るには少々心もとない。

出入りを見守る場合、本人の性格などが介入しない部分で見守る必要がある。一つの機械で出入りを判断するのであれば、超音波距離センサーが最適だと判断し、「超音波距離センサーを用いた見守りサービス」の研究を行い、評価を行う

2.3 超音波距離センサーとは

超音波距離センサーとは超音波の反射時間を用いて非接触で測距するモジュールである。外部からの入力で超音波パルス（8 波）が送信され、出力された反射時間信号を計算することで距離を測ることができる。

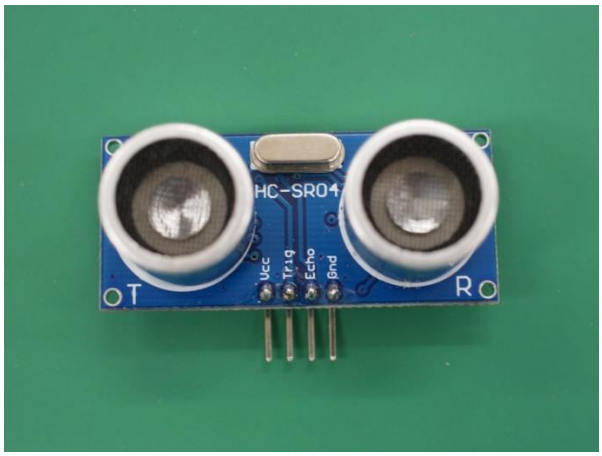

図 6　実験に使用した HC-SR04

第3章 提案手法

前章であげたようにドアの開閉状態によって見守り感知に影響がないようにするため、ドアより手前までを感知範囲とし、連続的な感知より、見守りの対象が距離センサーから離れていっているか近づいてきているかで外出と帰宅を感知できるようなシステムが必要になる。

3.1 提案手法

ドアに向かう方向で超音波距離センサーを設置する。見守り対象の老人が出ていく際、超音波距離センサーの前を通り、距離がログに残る。その距離の変化から外出と帰宅を判断する。超音波距離センサーは一定時間ごとに動作しているので、例えば最寄りのコンビニまで普段１時間で往復している以外でほとんど外出しない人なら、２時間以上帰宅の反応がない場合に見守る側の人にメールを送信し、異常を通達する。また、深夜や朝早すぎる時間などの外出も同様にメールを送ることによって徘徊の早期発見等に役立てることができる。そうして１日見守り、深夜０時に一度ログを保存しつつ出入りのまとめを送信し、普段通りの生活を送っているかも確認できるようにする。

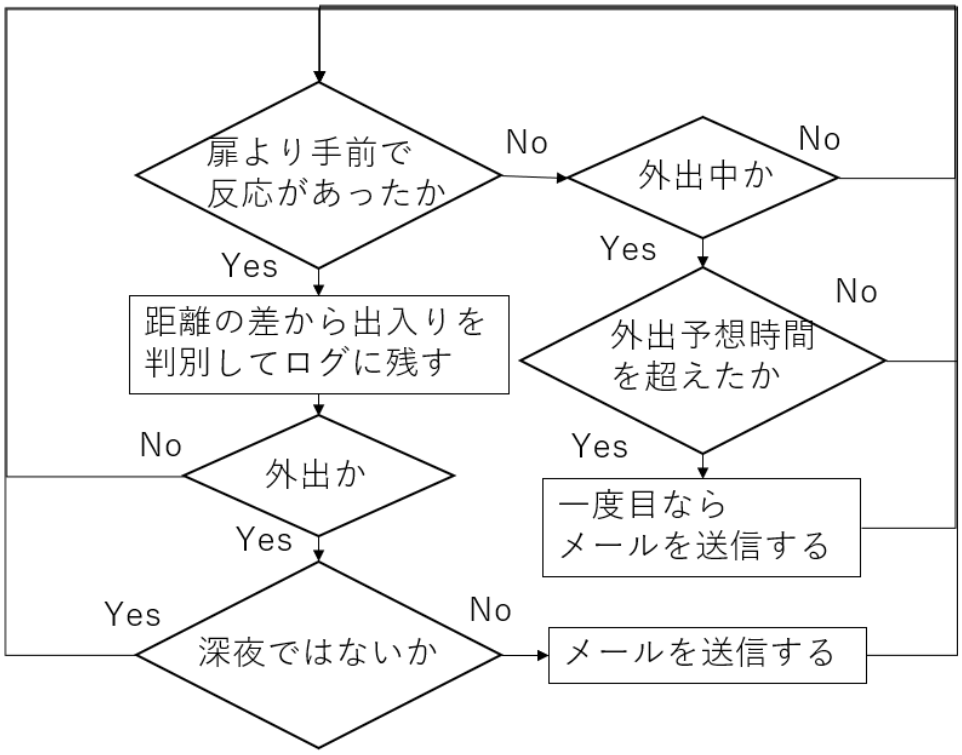

図７　提案手法フローチャート

第4章 実験

4.1 実験 1

4.1.1 実験目的

前章で提案した手法を用いて出入りの推測データと実際の外出時間を比較し精度を調べ、実際に一人暮らしの老人向けの見守り運用に向く精度であるかを判断する。また、どこに課題点があるか等も判断する。

4.1.2 実験概要

ドアに向かう方向で超音波距離センサーを設置する。最初にドアまでの距離を測定し、測定値より 5 センチ手前まで判定部分を設定する。0.3 秒間に 1 回測定を行い、設定された判定部分以遠の測定値が出る場合はそのまま測定を続ける。ドアとの間に人がいた場合は距離と時間をログに書き込む。

超音波距離センサーは埃等にも反応するので、一定時間内での反応数と距離の変化で出入りを推測する。外出して一定時間たっても帰ってこない、夜遅くに外出する等があればメールを送信する。

実験を行った対象にその日の外出時間を書き留めてもらい、ログと比較して精度を評価する。

今回、実験環境下では扉の向かいに成人の胸位の高さに小窓があったため、そこに超音波距離センサーとラズベリーパイを設置、固定する。超音波センサーは扉の鍵の部分を正面に向けるように調整しておく。

本実験で使用するラズベリーパイではリアルタイムでの処理は難しいと思われたので、あらかじめ取ったデータから一定時間以上の外出と夜遅くの外出、帰宅等を判断する。

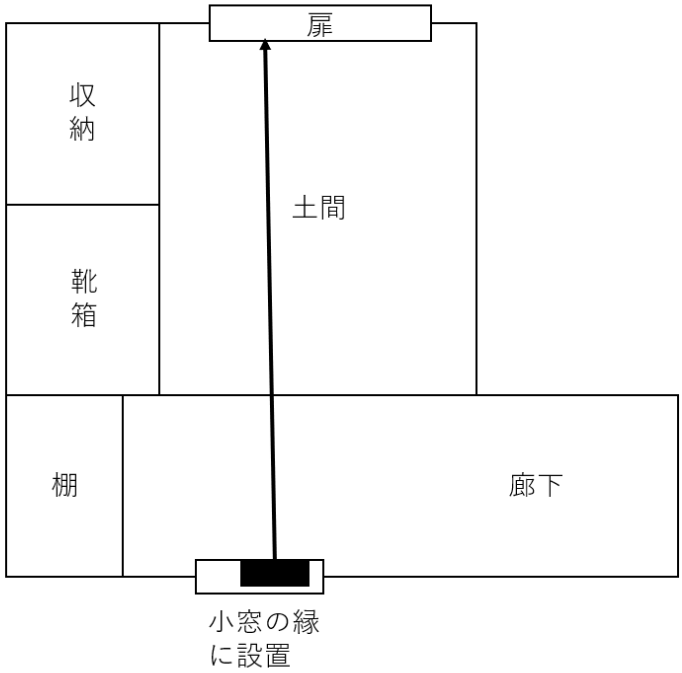

図 8　実験に使用した玄関の間取り

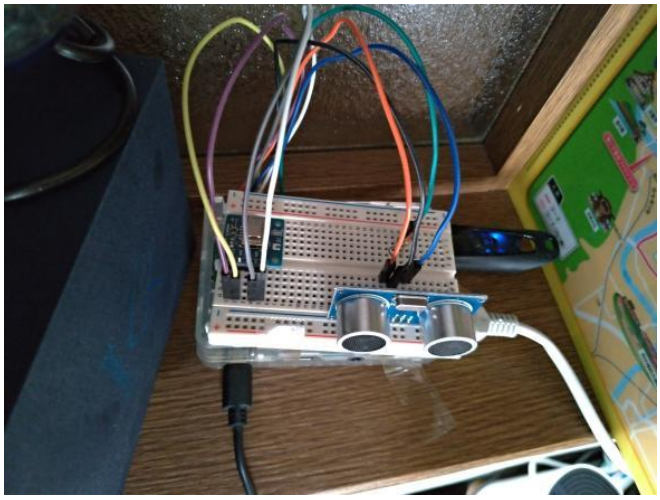

図 9　小窓に設置した超音波距離センサーとラズベリーパイ

4.1.3　実験方法

ラズベリーパイを超音波センサーと共に実験で使用し、固定するため、ラズベリーパイを有線 LAN でルーターと接続したうえで VNC Viewer を用いてリモートで操作を行い、Jupyter-Notebook からプログラムの実行､動作状態の管理を行う｡メールの送信には Gmail を使用し、アカウントを新規作成した。

超音波距離センサーで距離を測定する方法は[5]を参考とした。

4.1.4　システム概要

1、距離センサーにドアまでの距離を入力する。
2、ドアより 5 センチ手前までを感知範囲とし、センサーで判定を行う
3、ドアより近い距離で感知した場合、ログに記録する。
4、3 までを 0.3 秒の間隔をあけて連続で実行する。
5、1 日毎にログを保存する。
6、ログからデータフレームを作成する。
7、45 カウント以内に 3 回以上感知した部分を人間が通ったと判定し、距離の差から外出、帰宅を推測してメール用ログに記録する。
8、5 の中に、許容時間以上の外出と、夜間の外出の存在を調べ、存在した場合、別途メールを送信する。

4.1.5　結果

今回は、エラー等の理由により十分にデータが得られなかった日や、人の出入りが極端に少なく、精度に影響を与えかねない日を除いた計 13 日分のデータを採用する。

実際の出入りとメールで送信された出入りを比較し、正確に見れているもの、出入りの判定ミス、二重に感知したもの、感知されなかったもの、誤認感知の回数を表にまとめ、精度と目立つ感知ミスを算出する。

二重に感知したものの例としては同じ時間、1 分違いまでを二重感知とし、それ以降は誤認感知としてカウントする。実験環境の時計の大半がアナログ時計なため、手帳に書かれた時間はずれが発生していることが多いため、4 分以内の反応は正しいものとする。それ以上のずれは誤認感知とみなす。

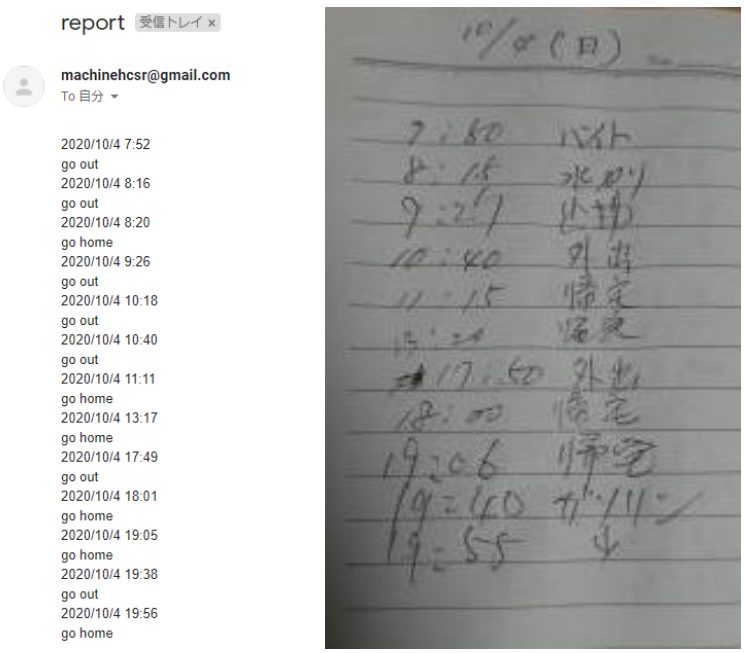

図 10　メールの文面と実際の外出時間を記入したもの

	9/15	9/22	9/23	9/24	9/28	9/29	10/1	10/2	10/3	10/4	10/5	10/11	10/12	合計
正	7	10	10	10	13	13	9	10	12	12	14	9	14	143
誤	2	3	3	8	13	5	1	3	2	1	7	2	4	54
精度	77.8%	76.9%	76.9%	55.6%	50.0%	72.2%	90.0%	76.9%	85.7%	92.3%	66.7%	81.8%	77.8%	72.6%

図 11　45 カウントでの精度

	9/15	9/22	9/23	9/24	9/28	9/29	10/1	10/2	10/3	10/4	10/5	10/11	10/12	合計
二重感知	0	2	0	2	1	1	0	0	2	0	0	1	2	11
誤認感知	2	1	1	3	10	1	0	1	0	1	5	0	0	25
未検知	0	0	0	2	0	2	0	1	0	0	1	0	0	6
出入違い	0	0	2	1	2	1	1	1	0	0	1	1	2	12

図 12　45 カウントでの誤感知の内訳

9/24、9/28、10/5 の精度が低いのが目立つ。この日はいずれも、ゴミ出しや玄関掃除、郵便、宅配など、多めの買い物による往復動作など短時間での玄関の出入りが多かった。これにより、出入り以外での感知や、二重に感知することがあり、短時間での出入りに弱いことがわかる。次に判定に使うカウント幅を狭くして、短時間の出入りへの精度の違いを比較してみる。

	9/15	9/16	9/17	9/18	9/19	9/20	9/21	9/22	9/23	9/24	9/25	9/26	9/27	合計
正	7	10	10	12	16	13	8	8	12	12	14	9	15	146
誤	2	3	2	6	11	7	3	6	2	1	11	1	4	59
精度	77.8%	76.9%	83.3%	66.7%	59.3%	65.0%	72.7%	57.1%	85.7%	92.3%	56.0%	90.0%	78.9%	71.2%

図 13　30 カウントでの精度

	9/15	9/16	9/17	9/18	9/19	9/20	9/21	9/22	9/23	9/24	9/25	9/26	9/27	合計
二重検知	0	2	0	1	2	3	1	2	2	0	5	0	3	21
誤検知	2	1	0	2	8	2	0	1	0	1	6	0	1	24
未検知	0	0	0	0	0	2	0	1	0	0	0	0	0	3
出入違い	0	0	2	1	1	0	2	2	0	0	0	1	0	9

図 14　30 カウントでの誤感知の内訳

未検知の回数は減ったが、特に二重検知の増加が目立つようになった。精度も上がっているところはあるが全体でみると下がっている。しかし、精度は 70%を保っており、一定期間での感知数で出入りは見れているだろう。

発生したエラーは出した超音波が返ってくるまでの時間の変数が空になり止まる事例と、プログラムの動作に使用している。Jupyter-Notebook やラズベリーパイ自体がフリーズする事例の 2 つがほとんどだった。

4.2 実験２

4.2.1 実験目的

前項の実験の際、処理を行う上でのエラーは起こったが、発生のタイミングがまちまちで、2 日以上の長時間稼働もできている日もあったため、リアルタイムでの処理も動作するか実験を行い確認する。

4.2.2 実験概要

前項で行った動作をリアルタイムでも同様の動きをできるようにする。なお、実験１と環境は同じままで行う。

データフレームを使用した判定は行えないので、１カウントごとに条件分岐で人の判定や出入りの判定を行うことになる。

１日の終わりでログを保存、新しいログを作成し、１日の出入りをメールで送信する。
一定時間以上の外出や、夜遅くの外出はその時点でメールを送信する。
実験１と同じく、実際の外出時間を書き留めてもらい、精度を評価する。

4.2.3 実験方法

実験１と同じ方法で行う。

4.2.4 システム概要

１、ドアまでの距離、夜間外出の期間、外出予想時間を入力する。
２、ドアより近い距離で感知した場合、感知ログに記録する。
３、一定カウント以内で再び感知した場合、最初の感知距離を保存し、感知が途切れるまで待機する。
４、感知が途切れた時、最初と最後の感知距離から外出、帰宅を判別する。
５、外出、帰宅を報告用ログに時間と一緒に記録する。
６、外出予想時間を超えても帰宅がない、夜遅くに外出した場合、メールを送信する。
７、1 日の終わりに報告用ログをメールで送信する。
８、感知ログを保存し、新しい感知ログを作成する。

今回も前回の実験と同じ 30-45 カウントの範囲で人を判別しようとしたが、1 サイクルの処理が前回の実験よりも多くなり。１カウントあたりの時間が大幅に伸びた。そのため今回はカウントの幅を 20 まで減らして実験を行う。

4.2.5 結果

実験中、前回ではほとんど見られなかった反響したと思われる反応が多数発生した。
反響の誤差は 3 センチ前後だったため、出入りの差が 10 センチ未満だった場合は反響と仮定し、除外するようにした。

また、反響による処理の連続でエラーが出ることが多くなり、実験結果として載せられる部分もかなり少なくなった。

	12/27	12/28	12/29	12/31	1/1	1/2	合計
正	5	7	6	7	7	7	39
誤	5	8	4	4	7	5	33
精度	50.0%	46.7%	60.0%	63.6%	50.0%	58.3%	54.2%

図 15　精度

	12/27	12/28	12/29	12/31	1/1	1/2	合計
二重検知	2	3	2	2	0	4	13
誤認検知	0	4	1	2	0	0	7
未検知	0	0	1	0	0	0	1
出入違い	3	1	0	0	7	1	12

図 16　誤感知の内訳

全体的に精度が低くなった。誤感知の内訳も出入り違いが増え、反響と思われる反応の影響が色濃くでる結果となった。特に、反響を感知した直後に人が通ると、反響位置の場所からの判定になってしまい、出入りが正しく判定できないことが問題となっている。

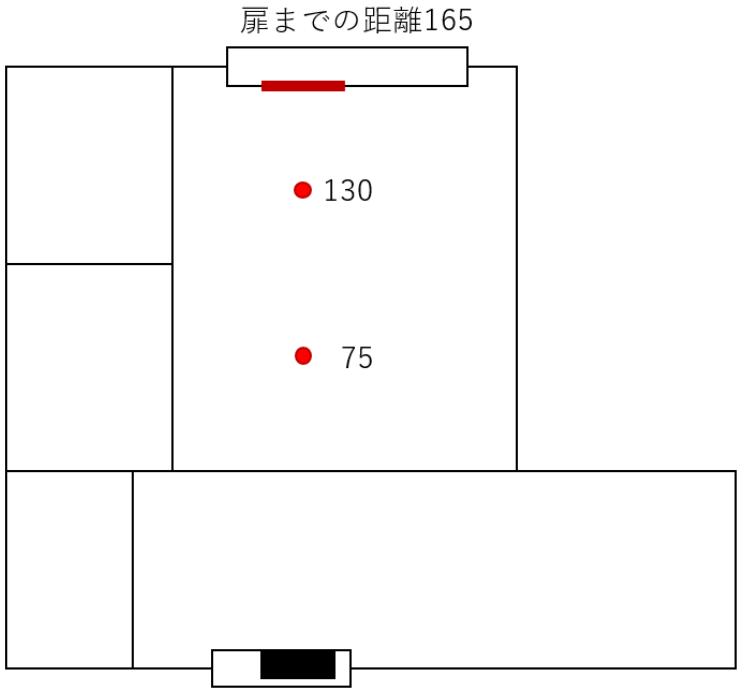

図 17　反響したと思われる反応の多かった位置

第5章　考察

実験１ではほこりの誤感知は予想していたよりはかなり少なかったが、水やりや買い物帰りで、荷物を取りに往復するなど、細かく往復した際にうまく途切れず、帰宅や外出が連続する現象が目立った。これらの解消のためにカウントの幅を狭めてみたが、誤感知や出入り違いは減少した代わり、二重感知がふえてしまった。両方とも条件分岐のみの絞り込みとしては 70%台と予想を上回る結果となったが、これ以上精度を上げるならば機械学習などを視野に入れる必要が出てくるだろう。例えば、二重感知を補正し、この時間帯の反応は大半が外出である等、特徴が出る部分は機械学習で修正できるだろう。

次に実験２では実験１ではほとんど見られなかった反響したと思われる反応が多数発生した。気温の影響なのか実験１以降に物が増えたのかが判別できなかった。実験の合間で反響を防ぐために、超音波距離センサーに影響がない範囲で遮蔽物を設置してみたが、頻度が少し減るだけでなくなることはなかった。また、反響の頻度もばらばらであり、距離も大半は前章での図の辺りだが、ブレ幅が大きいものもあり、玄関のどの部分が問題なのかもわからない状態だった。また、この反響もずっと続くわけではなく、自然と拾わなくなったり、突然発生したりと他の電化製品の干渉が原因とも思えない状態である。

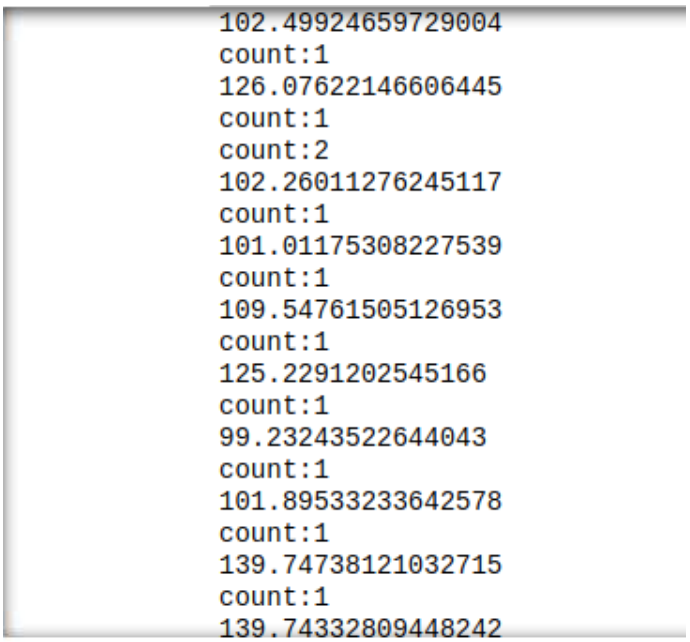

図 18　反響していると思われる状態（0:44 頃）

図 19　反響が原因だと思われる夜間外出反応

この反響を実験機材の不具合と仮定し、超音波距離センサーからブレッドボード等を取り換え、再び実験を行ったが、実験結果に変化は見られなかった。超音波距離センサーで見守りを行うならば、反響を取り除く条件分岐を実装するか、反響の誤受信を防ぐよう超音波距離センサーの周りをふさぐことが必要となってくるだろう。

第6章 まとめ

本研究ではカメラを使った見守りが嫌な一人暮らしの老人向けのカメラを使わない見守りサービスについて研究した。超音波距離センサーHC-SR04 をラズベリーパイに搭載し、玄関の扉の正面に設置、固定する。扉より手前での感知をログに残し、連続で感知した場合は人と認識し、出入りを距離の差から推測するプログラムを作成した。その際、普段の外出時間の平均時間と夜間外出の期間を設定しておけば外出して一向に帰ってこない時や、夜遅くの外出の時に指定されたアドレスにメールを送信し、見守り対象の異変を通知する機能を実装した。1日の終わりに出入りの報告も送られ、外出の頻度なども確認できる。

実際にこのプログラムを動かすにはリアルタイムでの処理が必要であり、ラズベリーパイでは処理しきれないように感じたため、まずはとったログから人の出入りと異常感知をおこなってみた。結果、精度は 70%程となった。誤感知や出入り違いはカウント幅の調整で減少したが、その代わり二重感知がふえてしまった。長時間稼働はできていたので、リアルタイムで処理を行う方でも実験を行ったが、反響と思われる反応が多数はいり、精度が 50%まで低下してしまった。超音波距離センサーの周囲に遮蔽物を置くと、頻度が少し減ったが、また違う距離からの反響と思われる反応が発生していた。初めの実験では反響と思われるような同じ距離からの連続的な反応はほとんど見られなかったため、実験環境の変化が考えられる。

今後の課題点として、反響への対策、機械学習の使用による二重感知や出入り違いの補正などがあげられる。また、メール送信に使用していたコマンドに文字コードが対応しておらず、日本語を使用できなかった。こちらの改善も必要だと考えている。

謝辞

本研究を進めるにあたり、ご指導を頂いた卒業論文指導教員の三好力教授に深く感謝致します。また三好研究室の皆様には、多くのご指摘を下さり感謝致します。

参考文献

[1] Microsoft Word - 02 29 結果の概要（世帯）
https://www.mhlw.go.jp/toukei/saikin/hw/k-tyosa/k-tyosa17/dl/02.pdf
[2] 商品について ｜ みまもりほっとライン ｜ 象印マホービン株式会社
https://www.mimamori.net/product/
[3] SENSOR 環境センサー / 開閉センサー / eHub（イーハブ）
https://linkjapan.co.jp/product/sensor/
[4] 史上最もシンプルな見守り・防犯デバイス[HelloLight(ハローライト)] | IoT 電球
https://hellolight.jp/
[5] 超音波距離センサ(HC-SR04)を使う - モノづくりサイト
http://make.bcde.jp/raspberrypi/%E8%B6%85%E9%9F%B3%E6%B3%A2%E8%B7%9D%E9%9B%A2%E3%82%BB%E3%83%B3%E3%82%B5hcsr04%E3%82%92%E4%BD%BF%E3%81%86/

付録

実験 1 で使用した、ログを残すプログラム

```
#hcsr04 を使用して距離を測定する。
def reading(sensor):
    import time
    import RPi.GPIO as GPIO
    GPIO.setwarnings(False)

    GPIO.setmode(GPIO.BOARD)
    TRIG = 11
    ECHO = 13

    if sensor == 0:
        GPIO.setup(TRIG,GPIO.OUT)
        GPIO.setup(ECHO,GPIO.IN)
        GPIO.output(TRIG, GPIO.LOW)
        time.sleep(0.3)

        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)

        while GPIO.input(ECHO) == 0:
          signaloff = time.time()

        while GPIO.input(ECHO) == 1:
          signalon = time.time()

        timepassed = signalon - signaloff
        distance = timepassed * 17000
        #print(distance)
        return distance
        GPIO.cleanup()
    else:
        print ("Incorrect usonic() function varible.")

print(reading(0))



import datetime
import time
#ドアまでの距離
DOOR = 150
#2 センチ以内で反応があればログを保存して終了する。
FIN = 2
#ログの作成
log = open('log.txt','a')
while True:
    DISTANCE = reading(0)
    #トアの手前に何かがあればログを残す
    if DISTANCE < DOOR:
        now = str(datetime.datetime.now())
        log.write(now)
        print(now)
        log.write('¥n')
        log.write(str(DISTANCE))
        log.write('¥n')
        print(DISTANCE)
    if DISTANCE < FIN:
        log.close()
        break

    time.sleep(0.3)
```

実験 1 で使用する出入り判別用にログのフォーマットを整えるプログラム

```
#日付と行数を指定
txt = '0918'
lines = 778

f = open ('log/' + txt + 'log.txt','r')
w = open ('trans/' + txt + 'transedlog.txt', 'x')

datalist = f.readlines()
i = 0
for i in range(lines):
    if i % 2 == 0:
        w.writelines(datalist[i].rstrip('¥n') + ",")
        continue
    w.write(datalist[i])

f.close()
w.close()

lines = int(lines) / 2
r = open('trans/' + txt + 'transedlog.txt','r')
log = open('format/' + txt + 'format.csv','x')

datalist = r.readlines()
for i in range(int(lines)):
    day = datalist[i][0:19]
    h = datalist[i][11:13]
    m = datalist[i][14:16]
    s = datalist[i][17:19]
    t = int(h) * 3600 + int(m) * 60 + int(s)
    l = datalist[i][27:32]
    index = datalist[i].find("hit")
    if index != -1:
        d = 1
    else:
        d = 0
    log.write(str(t) + "," + str(l) +"," + str(d) + ",0," +
str(day)+ "¥n")
r.close()
log.close()
```

実験 1 で使用した。ログから出入りを判別したプログラム

```
#ライブラリのインポート
import pandas as pd
import matplotlib.pyplot as plt
```

```
import smtplib

#データセットを読み込む

df_wine_all=pd.read_csv('format/0918format.csv', header=None)

df_wine=df_wine_all[[2,0,1,3,4]]
df_wine.columns = [u'class',u'time',u'distance',u'date',u'day']
pd.DataFrame(df_wine)
#プロットしてみる
%matplotlib inline

x=df_wine["time"]
y=df_wine["distance"]
z=df_wine["class"]-1
plt.scatter(x,y, c=z)
plt.show

#メール送信の準備
smtp_host = 'smtp.gmail.com'
smtp_port = 465
username = 'machinehcsr@gmail.com'
password = 'ryukoku04'
from_address = 'machinehcsr@gmail.com'
to_address = 'machinehcsr@gmail.com'
subject = 'report'
mailbody = '¥n'

smtp = smtplib.SMTP_SSL(smtp_host, smtp_port)
smtp.login(username, password)

period = 1
#1 日のカウント数
clock = 86340
#一定期間
term = 30
#判定用
flag = 0
#日付の判別用
setdate = 0
#外出時間の一時保存
outtime = -1
#一定時間の設定
ah = 2
am = 0
#夜間外出の設定
nh = 20
nm = 0
#メール用テンプレート
longout = 'Going out too long'
nightout = 'Going out late at night'
#一定時間、夜間外出をカウント数に
allowout = ah * 3600 + am * 60
allownightout = nh * 3600 + nm * 60

while period < 86340:
    if (outtime != -1 and period - outtime > allowout):
        print("long outing")
        lout = lday +"¥n" + longout
        wmessage = ("From: %s¥r¥nTo: %s¥r¥nSubject: %s¥r¥n¥r¥n%s" % (from_address, to_address, subject, lout))
        result = smtp.sendmail(from_address, to_address, wmessage)
        outtime = -1
    if (((df_wine["time"] >= period) & (df_wine["time"] < period + term) & (df_wine["date"] == setdate)).sum() >= 3):
        if (flag == 0):
            start = df_wine[(df_wine["time"] >= period) & (df_wine["time"] < period + term) & (df_wine["date"] == setdate)].iloc[0]["distance"]
            print (start)
            flag = 1
    else:
        if (flag == 1):
            end = df_wine[(df_wine["time"] >= period) & (df_wine["time"] < period + term) & (df_wine["date"] == setdate)].iloc[-1]["distance"]
            daytime = df_wine[(df_wine["time"] >= period) & (df_wine["time"] < period + term) & (df_wine["date"] == setdate)].iloc[-1]["day"]
            print(end)
            print(daytime)
            mailbody += daytime
            if (end - start > 0):
                print ("go out")
                mailbody += '¥ngo out¥n'
                lday = daytime
                outtime = df_wine[(df_wine["time"] >= period) & (df_wine["time"] < period + term) & (df_wine["date"] == setdate)].iloc[-1]["time"]
                print (outtime)
                if (period > allownightout):
                    print("night outing")
                    nout = daytime + "¥n" + nightout
                    nmessage = ("From: %s¥r¥nTo: %s¥r¥nSubject: %s¥r¥n¥r¥n%s" % (from_address, to_address, subject, nout))
                    result = smtp.sendmail(from_address, to_address, nmessage)
            elif(end - start < 0):
                print ("go home")
                mailbody += '¥ngo home¥n'
                outtime = -1
            else:
                print("just passed")
                mailbody += '¥njust passed¥n'
            flag = 0
    period += 1
```

実験 2 で使用したリアルタイム用のプログラム

```
#ライブラリのインポート
```

```
import time
import smtplib
import datetime

#hcsr04 を使用して距離を測定する。
def reading(sensor):
    import time
    import RPi.GPIO as GPIO
    GPIO.setwarnings(False)

    GPIO.setmode(GPIO.BOARD)
    TRIG = 11
    ECHO = 13

    if sensor == 0:
        GPIO.setup(TRIG,GPIO.OUT)
        GPIO.setup(ECHO,GPIO.IN)
        GPIO.output(TRIG, GPIO.LOW)
        time.sleep(0.3)

        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)

        while GPIO.input(ECHO) == 0:
          signaloff = time.time()

        while GPIO.input(ECHO) == 1:
          signalon = time.time()

        timepassed = signalon - signaloff
        distance = timepassed * 17000
        #print(distance)
        return distance
        GPIO.cleanup()
    else:
        print ("Incorrect usonic() function varible.")

print(reading(0))

#メール送信の準備
smtp_host = 'smtp.gmail.com'
smtp_port = 465
username = 'machinehcsr@gmail.com'
password = 'ryukoku04'
from_address = 'machinehcsr@gmail.com'
to_address = 'machinehcsr@gmail.com'
subject = 'report'
Longout = 'Going out too long'
Nightout = 'Going out late at night'
mailbody = 'report¥n'

#ログイン
smtp = smtplib.SMTP_SSL(smtp_host, smtp_port)
smtp.login(username, password)

#時刻から、カウント用に変換する
def trans()：
    h = now[11:13]
    m = now[14:16]
    s = now[17:19]
    t = int(h) * 3600 + int(m) * 60 + int(s)
    return (int(t))

#ドアまでの距離
DOOR = 150
#許容外出時間
ah = 2
am = 0
longout = ah * 3600 + am * 60
#夜間外出閾値
nh = 20
nm = 0
nightout = nh *3600 + nm * 60
#早朝外出閾値
mh = 6
mm = 0
morningout = mh *3600 + mm * 60
#ログの作成
log = open('log' + str(datetime.datetime.now())[0:10] +
'.txt','a')
#感知用変数
state = 0
#カウント数
count = 0
#外出判定用変数
out = 0
#外出カウント
outsec = -1
while True:
    DISTANCE = reading(0)
    #トアの手前に何かがあればログを残す
    now = str(datetime.datetime.now())
    sec = trans()
    if DISTANCE < DOOR:
        log.write(now + "," + str(DISTANCE) + "¥n")
        print (DISTANCE)
        if (state == 0):
            #"一度目の検知でカウントを始める
            print("state:1")
            state = 1
            count = 0
        elif (state == 1):
            #20 カウント以内に新たに判定があれば距離
を保管して判定が途切れるまで待つ
            if(count < 20):
                print("state:2")
                state = 2
                start = int(DISTANCE)
                end = int(DISTANCE)
                count = 0
            else:
                print("state:0")
                count = 0
                state = 0
```

```
        elif (state == 2):
            if(count < 20):
                #20 カウント以内に反応があれば継続
                count = 0
                end = int(DISTANCE)
    if(state == 1 and count >= 20):
        state = 0
        print("state:0")
        count = 0
    if(state == 2 and count >= 20):
        #20 カウント経過しても反応がない場合終端をとり、出入りを判断する。
        if(end - start > 10):
            #外出をメールのログに書く
            print("go out")
            mailbody += str(now)
            mailbody += '¥ngo out¥n'
            if (sec >= nightout or sec < morningout):
                #一定時間以降の外出を通知するメールを送る
                print("nightout")
                nout = str(now) + '¥n' + Nightout
                smtp = smtplib.SMTP_SSL(smtp_host, smtp_port)
                smtp.login(username, password)
                nmessage = ("From: %s¥r¥nTo: %s¥r¥nSubject: %s¥r¥n¥r¥n%s" % (from_address, to_address, subject, nout))
                result = smtp.sendmail(from_address, to_address, nmessage)
            #外出の時間を取る
            out = 1
            outtime = now
            outsec = sec
            print("state:0")
            state = 0
            count = 0
        elif(end - start < -10):
            #帰宅をメールのログに書く
            print("go home")
            mailbody += str(now)
            mailbody += '¥ngo home¥n'
            out = 0
            print("state:0")
            state = 0
            count = 0
        else:
            print("state:0")
            state = 0
            count = 0

    if(out == 1 and sec - outsec >= longout):
        #長時間外出のメールを送る
        print("longout")
        lout = str(now) +'¥n' + Longout
        smtp = smtplib.SMTP_SSL(smtp_host, smtp_port)
        smtp.login(username, password)
        wmessage = ("From: %s¥r¥nTo: %s¥r¥nSubject: %s¥r¥n¥r¥n%s" % (from_address, to_address, subject, lout))
        result = smtp.sendmail(from_address, to_address, wmessage)
        out = 0
    count += 1
    if(sec < 20):
        print("log close")
        log.close()
        log = open('log' + str(datetime.datetime.now())[0:10] + '.txt','a')
        print("send mail")
        smtp = smtplib.SMTP_SSL(smtp_host, smtp_port)
        smtp.login(username, password)
        message = ("From: %s¥r¥nTo: %s¥r¥nSubject: %s¥r¥n¥r¥n%s" % (from_address, to_address, subject, mailbody))
        result = smtp.sendmail(from_address, to_address, message)
        print(result)
        mailbody = 'report¥n'
        time.sleep(21)
    if (state != 0):
        print("count:" + str(count))
    time.sleep(0.3)
```