Vol.2021-ICS-202 No.4
2021/3/29

D_Find Obj:深層学習による物体検出で物の追跡手法と物探し支援システム

加藤 綾規[1,a)]　中山 裕貴[2]　濱川 礼[1]

概要：本論文では深層学習を用いた物体検出で物を追跡して，物探しを支援するシステム「*D_Find Obj*」について述べる．無くしやすい物の例としてリモコンやスマートフォンがある．これらは使用位置が決まっておらず，使用後に置いたり，仕舞ったりする場所が分散しやすい．その結果，再び使用する際にどこに置いたのかを忘れてしまい物探しを行わなければならない．物探しという行為は日常的に行われており，多くの時間をかけている．そのため，世の中には，物探しの時間を減らすための商品も出ている．例えばタグを物に取り付けて追跡する商品があるが，探すもの全てにタグを取り付ける必要があること，タグを取り付けることで使いづらくなる，などのデメリットが存在する．そこで「*D_Find Obj*」では部屋全体の様子が分かるようにカメラを設置して映像から物を追跡することで，装置を取り付けるなどのユーザの手間を軽減し，物には何も付加することなく物探しの支援を可能にする．「*D_Find Obj*」は物探しを行う時にユーザが見て瞬時に物の場所が分かるように部屋全体の画像内に矩形を描画して場所を示す画像と検出されたときの人の様子が分かる動画をユーザに提供する．深層学習を用いた物体検出には学習したモデルが必要となるため，「*D_Find Obj*」を使用するユーザは事前に追跡して欲しい物を登録する．登録には追跡する物の名前と画像を入力する．モデルを作成する時に必要になる訓練データは大規模データセットのクラス名と入力したクラス名を参照してデータを取得した．また，物体検出を行う際にはユーザが追跡して欲しい物と異なる物を検出してしまうことがある．例えばリモコンを検出すると，テレビやエアコン，レコーダーなどの様々なリモコンを検出してしまう．そこで追跡して欲しい物の画像と検出した画像の類似度を求め閾値を取ることで誤検出を減らした．

1. 背景・目的

部屋のどこでも使えてしまう物，例えばリモコンなどは使用した後に決まった位置に戻さず使用した周囲に置いてしまうことがある．この結果，時間が経って再び使用する際に置いた位置を覚えておらず部屋中を探し回る場合がある．TrackR,Inc. 社が 2017 年に行ったインターネット調査「探し物に関する調査」[1] では，人が 1 年間に費やす物探しの時間は 145 時間であることが分かっている．

物探しの時間を削減するためには最後に使用した場所・状況を物探しをしている人に伝えることで探す場所を狭めることが有効である．最後に使った場所を確認できる商品がある [2][3]．これらは物に取り付けることで物がどこにあるのかを確認することができ，探す場所を狭めることができる．しかし，これらの商品は探したい物全てにタグを取り付ける必要があり非常に手間であることや探したい物の数が多くなるほどコストがかかってしまう．また，取り付ける物によっては装置が邪魔になり使い勝手が悪くなることもある．そこで，部屋にカメラを設置して物体検出を用いることで物に装置を取り付けることなく物の追跡を行う物探し支援システム「*D_Find Obj*」を開発した．

2. 提案手法

物探しが起こる要因の例として，使用前にはテレビ台に置いていたリモコンを人が使用後に元の位置に戻すことを面倒と感じて，元の位置に戻さずに自分の近くに置いてしまい，その結果，再び使用する際にリモコンの位置が分からなくなることが挙げられる．また，[4] の調査からも最後に物を置いた位置を忘れてしまう事が物探しの原因であるという結果が示されている．これらのことから，物探しの原因を人が物を最後に使用した位置を忘れてしまうことであると考えた．しかし，人を直接検出してしまうと腕を横に伸ばした時に手先が見切れてしまうことがあるため「*D_Find Obj*」では，人を検出するのではなく，物を動かしている人の手を検出してから物の追跡を行う．物を直接検出せずに手の検出を行っている理由は部屋全体の映像か

[1] 中京大学工学部情報工学科
[2] 中京大学大学院工学研究科情報工学専攻
[a)] ryo.kato0122@gmail.com

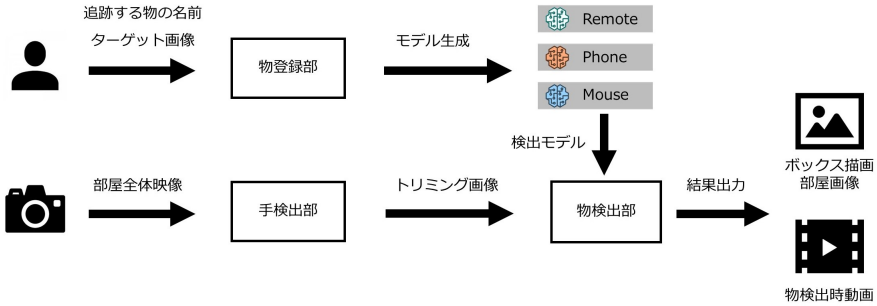

図 1 システムの流れ図

ら物を検出するよりも場所を絞って検出したほうが精度が向上するからである．物によってはカメラから離れてしまうと小さくなりすぎて検出の精度が低下してしまうが，手の場合ではカメラから離れていても検出できることが分かったため，物検出の入力を部屋全体の画像ではなく，手周辺を切り抜いて画像のサイズを小さくすることで検出の精度の向上を図るためである．また，部屋の上部にカメラを設置することで，俯瞰からの部屋全体の様子を取得した．

図 1 に「*D_Find Obj*」の流れ図を示す．「*D_Find Obj*」は追跡する物の登録を行う「物登録部」，部屋の全体の映像から手を検出する「手検出部」，手検出部で出力された手の周辺の画像内にユーザが登録した物を検出する「物検出部」の 3 つで構成する．ユーザには最後に検出された場所を矩形で位置を示した画像と，検出された時の前後数分の人の動きが分かる動画を提示する．

2.1 物登録部

物登録部出は，追跡する物を登録し物体検出を行うために必要なモデルの生成を行う．

ユーザには追跡する物の名前と追跡する物の画像を 1 枚を撮り，これらを入力し物登録を行う．一般的に物体検出のモデルを生成する際には多くの画像データが必要である．さらに全ての画像に対してどこに物があるのかを示すラベルを付与しなければならない．ユーザが追跡する物の画像を撮って学習データにすると，これらの作業を行う必要がありユーザに多くの負担をかけてしまう．そのため「*D_Find Obj*」では，既存の大規模な画像データベースを使用して追跡する物の名前と大規模な画像データベースに登録されているクラス名を照合しデータを取得して追跡する物ごとに検出モデルを生成する．追跡する物の画像は節 2.3 で後述する「物検出部」で使用する．また，大規模な画像データベースに例えば「mouse」と入力して照合すると PC の 「mouse」と動物の「mouse」の 2 種類のデータを取得してしまう．他にも「phone」を入力すると Mobile Phone と Telephone が一致と判定されてしまう．ユーザがコンピュータのマウスを追跡したい時は，動物のマウスのクラスのデータは必要ない．そこで，少数のデータをダウンロードして，ユーザが追跡したい物と近い物を含んでいるクラスデータを選択させるステップを経てから選ばれたクラスのデータを取得して学習データとした．

2.2 手検出部

手検出部では事前に手を検出するモデルを生成し，部屋全体の映像から人の手を検出して手の周辺のトリミングを行う．

2.3 物検出部

手検出部ではトリミングされた手周辺の画像に対して物登録部で生成された検出モデルを使いトリミングされた画像内に物があるのかを検出する．

図 3 と図 5 はユーザが登録したリモコンを検出例である．ユーザがテレビのリモコンを追跡して欲しかった時，図 3 は目的通りに検出することができているが図 5 は異なる物を検出している．誤検出してしまったものを削除するために検出結果とユーザが入力した追跡して欲しい物の画像を比較して類似度を算出する．算出した値に対して閾値を取り，誤検出の判定を行う．最後に検出した場所を矩形で囲った画像と検出した時の前後数分の人の様子が分かる動画を保存し，ユーザに提供する．

図 2 手検出結果

図 3 リモコンの正検出

図 4 手検出結果

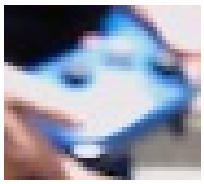

図 5 リモコンの誤検出

3. 関連研究

物探しの支援を行う研究はタグやカメラを使用するものなど今日まで様々なものが行われている．

3.1 物にタグを付与して物探し支援

物に RFID を取りつけておくことで位置を追跡して物探しの支援を行う研究がある [5]．部屋に Active RFID と超音波位置測定器を設置することでタグの位置を追跡し，ムービングライトの光を使って物の位置をユーザに伝える．しかし，探すものが増えるとタグが複数個必要があることやタグを取りつけることで物の使い勝手が悪くなってしまうことがある．「*D_Find Obj*」では物探しの支援にカメラを使用し，タグを使わないことでこれらのデメリットを解消している．

3.2 ウェラブルカメラを用いた物探し支援

ユーザにウェアラブルカメラを取り付けて物の使用位置を追跡して物探しの支援を行う研究がある [6][7]．[6] は頭上に，[7] は胸ポケットにカメラを付けることで，手の周辺の様子から物を使った時の状態を取得することができる．カメラを取りつけることで，部屋の中という限られた場所以外でも物の追跡ができるというメリットはあるが，常にカメラを装着する必要がありユーザに負担をかけてしまう．「*D_Find Obj*」ではカメラを部屋に設置することで，ユーザに直接負担をかけること無く物探し支援を行う．

3.3 部屋にカメラを設置して物探し支援

カメラを部屋に設置することで物の動きを追跡して物探しを行う研究がある [8][9]．[8] は，特定の引き出しの上部にカメラを設置することで，物を出し入れした様子を取得しユーザは引き出しのどこに何があるのかを web アプリケーションで確認することができる．[9] は部屋に Kinect センサを設置し深度画像を取得し，人の入出前後で差分を取ることで新しく現れたもの，消えた物の検出をすることで物探しの支援をしている．しかし，[8] では引き出しでしか物の追跡ができず，[9] では物が無くなった時の人の様子が確認できず詳細な場所を確認することが出来ない．「*D_Find Obj*」では部屋全体の様子を取得することで場所を問わずに物を追跡することを可能にし，常に物の動きを追跡することで，物が動いた時の人の様子を確認することができる．

4. システム構成

「*D_Find Obj*」のシステム構成図を図 6 に示す．「*D_Find Obj*」は物登録部，手検出部，物検出部の 3 つのモジュールから構成される．

4.1 ユーザ入力

ユーザは以下の 2 つを入力することで追跡する物の登録を行う．

- 追跡する物の名前
- 追跡する物の画像

4.2 YOLO

「*D_Find Obj*」は物体検出に YOLO（You Only Look Once）[10] の後継モデルである YOLOv5[11] を使用している．YOLOv5 は深層学習による物体検出のモデルであり，他のモデルと比較して高速かつ高精度で検出が可能である．

また，YOLOv5 には学習データを読み込む際にデータ拡張を行う機能が搭載されている．入力された学習データに対して，拡大・縮小，回転，色変化等を加えて，画像データを拡張させている．このため，少ないデータで学習を行う時には他の学習モデルでは前処理でデータ拡張を行い過学習を防ぐが，YOLOv5 では学習を行う際に自動でデータ拡張を行うため，この前処理が必要ない．YOLOv5 は Pytorch を使用することで，学習やテストを行うことができる．

4.3 物登録部

物登録部ではユーザが追跡して欲しい物の名前と画像を受け取り，大規模な画像データベースである ImageNet と Open Images Dataset から訓練データの取得を行い，YOLOv5 で学習し検出モデルを生成する．モデルの生成は登録されたクラスごとに行う．

4.3.1 学習データの取得

入力された追跡する物の名前が事前に登録されていないかを最初に確認する．これは追跡する物の名前を使い ImageNet,Open Images Dataset から訓練データを取得する為，同一の名前では同じモデルを生成することになるからである．

入力された追跡する物の名前を使い ImageNet,Open Images Dataset のクラスリストを参照して一致したクラスの訓練データを 10 枚ダウンロードする．10 枚限定にした理由は入力したクラスによっては mouse と computer mouse のように同じ名前の物が存在するため，1 度ユーザには自分が登録したい物と近いクラスを選択した後に，選択されたクラスの訓練データをダウンロードする．ダウンロードには ImageNet，Open ImagesDataset のトレーニングデータ，バリデーションデータ，テストデータの枚数を上限 500 枚に設定した．これは，クラスによっては学習データがとても多くモデル生成にかかる時間が増えてしまうためである．テストデータは「*D_Find Obj*」では使うことがないため，バリデーションデータとして扱った．

ImageNet，Open Images Dataset のアノテーションは YOLOv5 では使用できない形式で記入されているため，

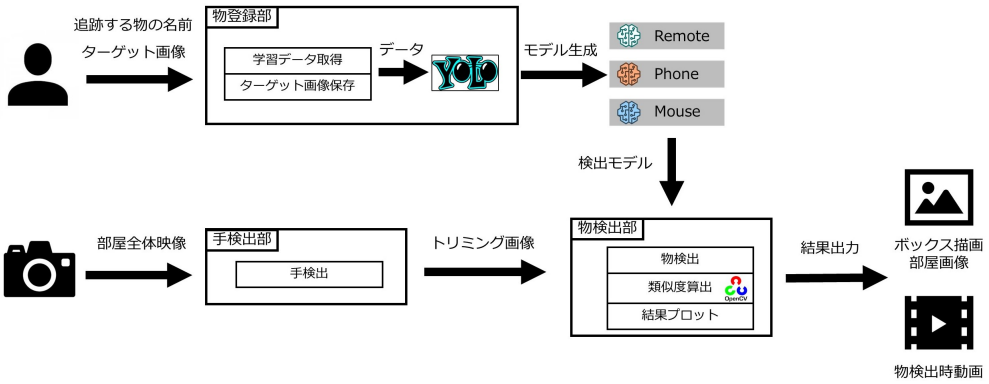

図 6 システム構成図

YOLOv5 で学習できるバウンディングボックスの値に変換し学習データとした．

4.3.2 ターゲット画像保存

物体検出のモデルを生成する時に，大規模データセットを使っているため目的の物に近い物も検出してしまう．そこで，検出された結果が目的の物か誤った物かを判別するために，ユーザが入力した画像を用いる．ユーザが入力した画像をターゲット画像とし，画像の名前を追跡する物の名前に変更して保存を行う．判別方法については節 4.5.2 で詳細に述べる．

4.3.3 物検出モデル生成

取得した訓練データを用いて物ごとに YOLOv5 の学習を行う．表 1 は YOLOv5 の事前学習済みモデルの性能を比較した表である．YOLOv5 では事前学習済みのモデルを YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x の 4 つの中から選んで学習することができる．これらのモデルはモデルの複雑さを表す params の値が異なっており，YOLOv5s が最も少なく YOLOv5x が最も多くなっている．また，表 1 から params の数が多くなるほど精度（AP^{test}）が高くなるが，処理にかかる時間は長くなってしまうことが分かる．表 2 は物登録部で remote と入力した時に取得した訓練データを使い学習にかかった時間である．訓練データのトレーニングデータは 465 枚，バリデーションデータは 140 枚であり，バッチサイズは 8，エポック数 300 で GPU（NVIDIA GeForce RTX 2060）を使い学習を行った．表 1 の AP^{test} の値と表 2 の Time の結果から YOLOv5m 以上ではモデル生成を行うたびに時間がかかり過ぎてしまう点と検出精度に大きな差が無い事が分かった．本システムではユーザが物を登録する度に物検出のモデル生成を行うため学習時間が長くなりすぎるとシステムとして使いにくくなる．そのため「*D_Find Obj*」では事前学習済みモデルは YOLOv5s を使い学習を行った．物検出モデル生成の学習に使用したパラメータは上記と同様でバッチサイズは 8，エポック数は 300 である．

	params	AP^{test}	Speed	FPS
YOLOv5s	7.5M	37.0	2.4ms	416
YOLOv5m	21.8M	44.3	3.4ms	294
YOLOv5l	47.8M	44.7	4.4ms	227
YOLOv5x	89.0M	49.2	6.9ms	145

表 1 学習済みモデルの比較

	Time
YOLOv5s	0.8h
YOLOv5m	1.5h
YOLOv5l	2.2h
YOLOv5x	4.2h

表 2 モデル生成にかかる時間

4.4 手検出部

手検出部では事前に生成したモデルを使用して，部屋全体の映像から人の手を検出し，検出した手の周辺をトリミングする．

4.4.1 手検出用データセット

検出モデルを生成するために必要なデータセットは Open Images Datasets からクラス名 Human hand で取得した．トレーニングデータ 22,093 枚, バリデーションデータ 1,676 枚である．節 4.3.1 と同様にアノテーションの形式を YOLOv5 で学習できるように変更し学習データとした．

4.4.2 手検出用モデル生成

節 4.4.1 の学習データを使い YOLOv5 で学習を行った．物検出モデルとは異なり手検出用のモデルは事前に生成するモデルであり，モデル生成の処理は何度も行われない．手検出の学習は，YOLOv5 の事前学習済みモデルの中で精度が最も高い YOLOv5x を使用した．学習の際に使用したパラメータはバッチサイズ 16 エポック数 300 で行った．

4.4.3 部屋全体映像

部屋の隅にカメラを設置して図 7 の様な部屋全体の様子が俯瞰から分かる映像を取得する．「$D - Find\ Obj$」で使用したカメラは Logicool 社の StreamCam(C9800W) である．このカメラ映像を使い物の位置の追跡を行う．物が動いた際に人の動きを確認できる動画を出力するためにカメ

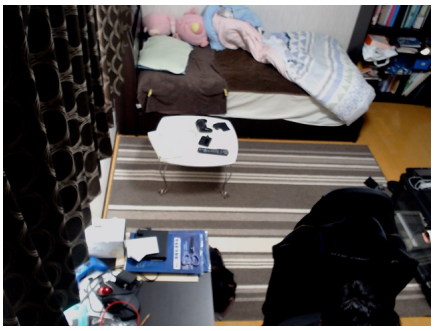

図 7 部屋全体映像の切り抜き

ラの映像を 1 分ごとに保存する．

4.4.4 手検出

YOLOv5 による検出は，毎秒 100 枚の画像を処理して検出することが可能であるが，動きが無い同じ様子の手を検出してしまう．この結果，手周辺の画像が増え，物検出部の処理が増えてしまう．このため「*D_Find Obj*」では取得する部屋全体映像に遅延をかけて手の検出を行う．こうすることで変化しない手を検出することを防ぎ，物検出での処理が増えてしまう問題を解決した．

また，YOLOv5 の物体検出ではバウンディングボックス内に対象のクラス物体が入っていて物体の領域を囲えているかの正確さを表す値 confidence がある．この値の閾値を取ることで正確性が低すぎる結果を排除できる．手検出を行う際の閾値は予備実験で精度が良かった値である 0.7 に設定して検出を行う．

手が検出された時には，検出されたバウンディングボックスより一回り大きいバウンディングボックスの座標と大きいバウンディングボックスでトリミングした画像，検出された時間を保存する．一回り大きいバウンディングボックスで保存する理由は検出されたバウンディングボックスでは物が手からはみ出した時に検出の精度が落ちてしまう事と，使っている時だけではなく手から少し離れた時も検出するためである．バウンディングボックスの座標と検出時間は節 4.5.3 で述べる結果を出力する際に使用する．

4.5 物検出部

物検出部では物登録部で生成した各クラスの検出モデルを使い，手検出部で作成した手周辺のトリミング画像で物体検出を行う．その後，検出した結果とターゲット画像の類似度を算出して誤検出かの判断を行い，ユーザに物の位置を示す画像と最後に物を使用した時の状況が分かる動画を出力する．

4.5.1 物検出

手検出部で手の周辺をトリミングした画像に対して節 4.3 でクラスごとに生成したモデルを使い物登録部で登録した物の検出を行う．検出の際に設定した confidence の閾値は手検出と同様の理由から 0.7 に設定した．

図 8 ターゲット画像

類似度算出を行うために，手検出部と同様に検出したバウンディングボックスに従いトリミングを行う．この時バウンディングボックスの座標と，トリミングされた手検出の何番目に物があったのかを示す番号を物ごとに分けてテキストファイルに記録する．

4.5.2 類似度算出

本システムの YOLOv5 による特定の物の検出は学習データを ImageNet，Open Images Dataset から作成してる為困難である．リモコンを検出すると図 3 と図 5 の様に種類の異なる物が検出されてしまう．そのため，図 8 の様にユーザが入力した追跡する物の画像（ターゲット画像）と検出された物との類似度を AKAZE[12] を使った特徴量と画像のカラーヒストグラムの比較を行い類似度を算出する．設定した閾値と算出した値を比較してターゲット画像と同じ物かを判定する．また，これらの類似度は OpenCV に組み込まれたモジュールを用いて算出している．

AKAZE

AKAZE での特徴量を求める上で画像の色の情報は不要であるため，比較する 2 つの画像をグレースケール変換した．AKAZE で算出した特徴量を opencv に組み込まれている BFMatcher(Brute-Force matcher) 関数を使用することで特徴点の距離が算出され 2 つの画像の類似度が数値として出力される．この数値が小さいほど一致していることが分かる．

カラーヒストグラム

通常カラー画像は RGB それぞれ 0〜255 の 256 通りの値で構成されるが，この値のままでは，特徴が多すぎるため僅かな差で類似度が低くなる．そこで入力する画像に対して減色処理を行ってからカラーヒストグラムを出した．比較する 2 つの画像のヒストグラムの値を Histgram Intersection と呼ばれる類似度を式 1 で RGB それぞれで算出する．式 1 の I は色の個数を H_1,H_2 は比較するヒストグラムを示している．算出した値は画像のカラーヒストグラムが近い程 1 近づく．RGB の中で最も値が大きかったものを使い，類似度の比較を行った．

$$d(H_1, H_2) = \frac{\sum_I min(H_1(I), H_2(I))}{\sum_I H_1(I)} \qquad (1)$$

4.5.3 結果プロット

誤検出を減らした物検出で書き込んだテキストには手検

出の何番目に物検出されたのかを示す番号と手検出でトリミングされた画像内のどこにあるのかを示す座標を保存している．さらに，手検出部でも検出された順に部屋全体のどこに手が検出されたのかを示す座標と，検出が開始されて何秒経っているのかが記入されている．これらの値を使い最後にユーザにどこにあるのかを表示する画像と，最後に検出された時の人の状況が分かる動画を出力する．

初めに画像について説明する．画像の検出を行う前にユーザには図 7 の様に部屋全体の写真を撮影する．図 9 は図 7 に矩形を描画してユーザ出力する例である．

次に動画について説明する．手検出部では 1 分ごとに動画を保存しており，手検出は 15 分行った後に物検出と進んでいくため，15 本の動画が作成される．また，手検出部で取得した画像を制限するためにディレイを設けたため，1 本当たりの動画は 7 秒程度になっている．手検出部では手を検出した時の時間を保存しているため「経過時間 /60」を行うことで最後に検出されたのが，何本目の動画であるのかが分かる．検出した時の動画と前後の動画を合成してユーザに出力する．

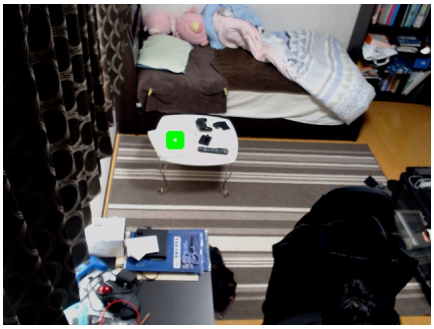

図 9 出力画像例

5. 評価実験

YOLOv5 を用いて学習させた手検出モデルと物登録部で「remote」と「phone」を入力して生成したモデルの性能評価と「*D_Find Obj*」の物の追跡精度の評価を行った．モデルの性能評価には物体検出の評価指標である mAP[13] の値を用いて評価した．物の追跡精度はある部屋内でリモコンとスマートフォンを移動させることで，実際に最後に物が置いてある地点と「*D_Find Obj*」の追跡結果を比較して評価をした．

5.1 モデルの性能評価

5.1.1 手検出モデル

節 4.4.2 で生成したモデルを使用して学習データに含んでいない 5,238 枚のテストデータに対して検出を行い，mAP を算出した．算出した結果を表 3 に示す．また，検出した結果を図 10，正解データを図 11 に示す．

閾値	Precision	Recall	mAP(%)
$mAP_{0.5}$	0.421	0.868	81.3
$mAP_{0.65}$	0.31	0.885	81.0
$mAP_{0.75}$	0.202	0.895	79.4
$mAP_{0.85}$	0.092	0.9	72.1

表 3 手検出のモデルの検出精度

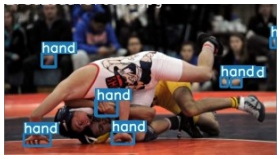

図 10 手検出結果

図 11 手検出正解データ

閾値	Precision	Recall	mAP(%)
$mAP_{0.5}$	0.872	0.931	94.5
$mAP_{0.65}$	0.854	0.931	94.4
$mAP_{0.75}$	0.839	0.949	94.2
$mAP_{0.85}$	0.736	0.951	92.9

表 4 remote のモデルの検出精度

表 3 の結果から検出の精度は $mAP_{0.75}$ で 79%の精度を達成することができたことが分かる．また，図 10 の様に背景の手などの，テストデータではアノテーションが付与されていない手を検出してしまっているために，間違っている物を検出していると判定されており，mAP の値が低下してしまっていると考えられる．

5.1.2 物登録部生成モデル

物登録部で生成するモデルを手検出の評価と同様に mAP を算出した．評価する検出モデルのクラスは remote（リモコン）と phone（スマートフォン）である．「*D – Find Obj*」ではテストデータをバリデーションデータとして扱っているが，評価に用いるモデルはバリデーションデータとテストデータを区別して学習を行った．

remote

トレーニングデータ 465 枚，バリデーションデータ 103 枚の学習データを使用し，エポック数 300，バッチサイズ 16 で検出モデルを生成した．このモデル使用して，学習データに含んでいないテストデータ 37 枚を使い mAP を算出した．算出結果を表 4 に示した．また，検出した結果を図 12，正解データを図 13 に示す．

表 4 から mAP の値はどの閾値であっても 90%の精度を達成できた．また，図 13 に注目すると，テレビの操作に用いるリモコンだけでは無く，ゲームに使うリモコンも検出していることから，汎化な検出モデルが生成されていると言える．

phone

トレーニングデータ 1,225 枚，バリデーションデータ 175 枚の学習データを使用し，remote と同様のパラメータで検出モデルを生成した．学習データに含んでいないテストデータ 374 枚を使い mAP を算出した．算出結果を 5 に

図 12 remote 検出結果

図 13 remote 正解データ

閾値	Precision	Recall	mAP(%)
$mAP_{0.5}$	0.744	0.641	71.2
$mAP_{0.65}$	0.736	0.645	70.6
$mAP_{0.75}$	0.713	0.645	68.6
$mAP_{0.85}$	0.641	0.645	64.6

表 5 phone のモデルの検出精度

図 14 phone 検出結果

図 15 phone 正解データ

示す．また，検出した結果を図 15，正解データを図 15 に示す．

表 5 から約 70%の精度であることが分かる．remote 比べ精度が大きく落ちている．これは phone の訓練データにはスマートフォンやガラケーなどの多くの形の物が入っており，学習データに制限を加えてしまった為学習の不足が起きたことが原因と考えられる．

5.1.3 考察

Open Images Dataset のから学習データを取得し，YOLOv5 で学習して作成した手検出モデルは良い精度結果であった．学習に用いたデータの数が多いこと，学習済みモデルである YOLOv5x を使用したことがこの結果に繋がったと考えられる．手を検出して，検出範囲を狭めるという目的が十分に達成することが出来るモデルであると言える．

追跡したい物を入力して，ImageNet と Open Images Dataset からデータを取得して生成したモデルは remote は高い精度であったが，phone は高い精度とは言えない結果であった．phone のデータには多くの形の物があり，「$D - Find\ Obj$」では学習時の時間をかけすぎないように，取得するデータに取得枚数の制限を加えてしまっている為学習が不足してしまっていると考えられる．また，remote の mAP はとても高い値であった．リモコンの形が phone に比べて変化がないため，十分に学習ができていると考えられる．また，テストデータの数が remote では phone に比べて少ないことも mAP の値が高い要因の 1 つであると考えられる．

5.2 追跡精度の評価

部屋を図 16 に示すように，場所に名前を付けた．物を名前を付けた場所に移動させて表示された結果と実際の位置が正しかったのかを判定する．remote の検出結果を表 5.2 に，phone の検出結果を表 5.2 に示した．

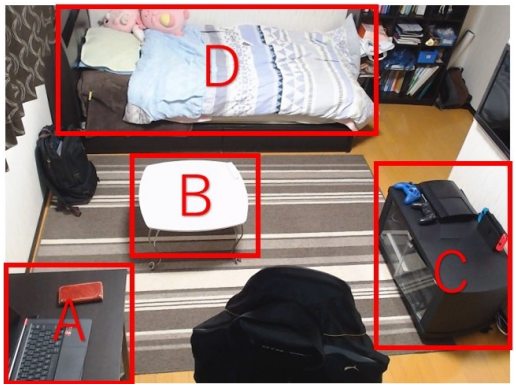

図 16 場所の名前

	A	A	B	C	D	D
result	○	○	○	○	×	×

表 6 remote の結果

	A	A	B	B	C	D
result	○	○	○	×	×	×

表 7 phone の結果

5.2.1 考察

remote，phone 共にカメラからの距離が最も近い位置である A 地点（約 2m）での物の追跡は誤検出がなかった．remote においては，最もカメラから離れている D 地点 (約 4m) 以外では追跡を行うことができた．

phone では，類似度の値からターゲット画像と検出物体が同じではないと判定された物体が，実際にはターゲット画像と検出物体が同じ物体であったケース（FN）が約 60%あった．図 17 に示した画像はスマートフォンを検出した結果を類似度判定でスマートフォンのターゲット画像とは同じではないと判定された例である．図 17 上段ではスマートフォンとは同じではないと正しく判定しているが，中段と下段の様にスマートフォンであるが，スマートフォンと同じではないと判定してしまっている．原因として，スマートフォンの画面は表示している画面や光の反射で大きく色が変わってしまい，カラーヒストグラムの類似度が小さくなるためと考えられる．また，remote は色の変化がないこと，大きさがスマートフォンに比べて大きいため誤検出が少なかったと考えられる．D 地点はカメラ映像から最も遠く物のサイズが小さくなってしまため，検出することが難しくなっていることが分かった．

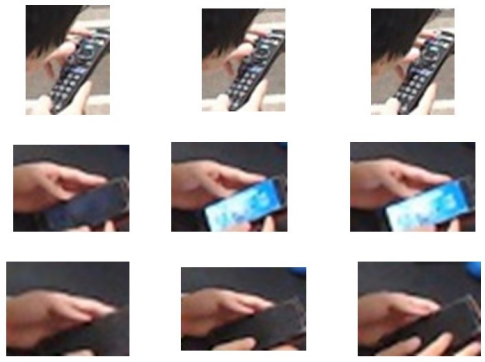

図 17 スマートフォンと異なる物と判定された画像

6. 展望

「*D_Find Obj*」ではカメラ映像に対して，深層学習を用いた物体検出を用いることで物の位置を追跡して物探しの支援を行った．

評価実験では，検出精度と誤検出の判定の精度が低いことが分かった．「*D_Find Obj*」は大規模データを用いることで，汎化なモデルを生成しているが，ユーザが登録したい物をスマホを使い 3D データとして読み取りユーザに負担がかからないように学習データにして物の追跡を行うことや，カラーヒストグラムと AKAZE 以外の方法で類似度の算出を行い精度の向上を図っていきたい．「*D_Find Obj*」を作動して行った評価実験では特定の部屋でしか評価を行っていないため，別の環境で長時間「*D_Find Obj*」を作動しシステムの精度をより正確に確かめる必要がある．「*D_Find Obj*」ではカメラを 1 台で物の追跡を行っているため，死角増えてしまっている．そのため複数のカメラを用いて検出を行うことが出来るように改良し，死角を減らす必要がある．また，カメラを設置して部屋の様子を取得しているため，プライバシーを守る必要があるが，現状ではユーザがカメラに映りたくない時にはカメラを OFF にする機能しか設けてない．そのため，他にもプライバシーを守るシステムを組み込む必要がある．

参考文献

[1] 探し物に関する調査　2021 年 3 月 4 日アクセス https://prtimes.jp/main/html/rd/p/000000006.000022312.html.
[2] Tile 公式サイト 2021 年 3 月 4 日アクセス https://bit.ly/35rBRuW.
[3] Pixie 公式サイト 2021 年 3 月 4 日アクセス https://web.archive.org/web/20171223030029/https://getpixie.com/.
[4] The Nation’s Biggest Lost and Found Survey, by Pixie　2021 年 3 月 4 日アクセス　https://tinyurl.com/yxrzbsnp.
[5] 中田 豊久，金井 秀明，國藤 進:スポットライトを用いた屋内での捜し物発見支援システム，情報処理学会論文 Vol.48 No.12（2007）．
[6] 上岡 隆宏，河村 竜幸，河野 恭之，木戸出 正継:物探しを効率化するウェアラブルシステム，ヒューマンインタフェース学会論文誌 vol.6，No.3(2002)．
[7] Cristian Reyes, Eva Mohedano, Kevin McGuinness, Noel E. O’Connor，Xavier Giro-i-Nieto”Where is my Phone ? Personal Object Retrieval from Egocentric Images”，Lifelogging Tools and Applications Workshop (LTA’16) at ACM Multimedia(2017)．
[8] 小松崎 瑞穂，塚田 浩二，椎尾 一郎:DrawerFinder:収納箱に適した物探し支援システムの提案と運用，情報処理学会第 73 回全国大会講演論文集, pp. 1–603–604,(2011)．
[9] 栗原 竜矢，岡部 誠，尾内 理紀夫:kinect センサを用いた物探し支援システムの試作,WISS デモ発表 (2011)．
[10] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi ”You Only Look Once: Unified, Real-Time Object Detection”，arXiv:1506.02640，2015．
[11] YOLOv5　github 2021 年 3 月 4 日アクセス https://github.com/ultralytics/yolov5
[12] P. F. Alcantarilla，J. Nuevo，A. Bartoli”Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces”，British Machine Vision Conf. (BMVC)，2013
[13] M. Everingham，L. Van Gool，C. K. I. Williams，J. Winn，A. Zisserman”The pascal visual object classes challenge 2012 (voc2012)”.2021 年 3 月 4 日アクセス http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html