B-05

モバイル端末を用いた複数物体の検出と SLAM による被災家屋の位置特定

Identification of the location of damaged houses by detecting multiple objects and SLAM using mobile devices

岸川 雄星†　河合 紀彦†　鈴木 基之†　伊勢 正‡
Yusei Kishikawa　Norihiko Kawai　Motoyuki Suzuki　Tadashi Ise

1. はじめに

災害により被害が出た場合，被害の状況を調査し，被災者を支援するための判断材料となる被害の程度を証明する書面である罹災証明書を交付する必要がある．具体的な被災から支援措置活用までの流れを図 1 に示す．

現在の被害程度の調査は，一部でモバイル端末の活用がみられるが，被害程度の調査の大部分が手作業で行われており，調査者が家屋を一軒ずつ移動しながら目視することで，その家屋の被害程度と位置を記録している．大きな災害の場合，確認すべき家屋の数が膨大になり，またその位置を GIS（地理情報システム）データ内の家屋とマッチングする必要があるため，作業に非常に多くの時間と労働力が必要となる[1].

そこで本研究では，この作業の効率化を目指して，遠方からモバイル端末のカメラ映像を用いて，複数の家屋を同時に検出・トラッキングすることで複数の家屋の位置を同時に特定するシステムを提案する．このシステムでは，Visual SLAM によるカメラ位置姿勢推定を利用する．このシステムにより，被害発生後の調査の一部に関して作業の時間と労働力の削減を目指す.

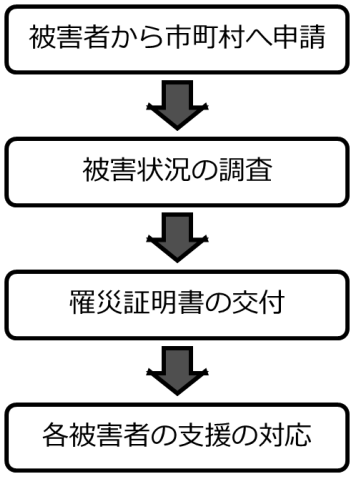


図 1　被災から支援措置活用までの流れ

2. 従来研究

特定の物体に限らない画像を用いた遠隔からの位置計測として，ステレオ法や Structure-from-Motion（SfM）法が提案されている[2]．しかし，これらは画像内から画像間で対応付けやすい特徴点を抽出し，その 3 次元位置を計測しているため，特定の物体の位置を対象としていない．一方で，物体検出と合わせることで，特定物体の位置を計測する手法が提案されている[3]．しかし，現状の多くのシステムではステレオ法により近くの物体の位置を特定することを目的としており，遠隔の物体を対象としていない．

また，別の方法として，拡張現実を利用した対象物体の位置特定がある．Slider[4]では，現実物体に対して注釈などの仮想物体を配置するため，まず 1 視点から対象物を指定する．次に，カメラを動かしたとき，最初の視点から対象物を結ぶ直線がエピポーラ線として画面に表示され，エピポーラ線上で仮想物体を動かすことで，仮想物体を現実物体の位置に配置することができる．このとき，副次的に仮想物体の 3 次元位置がわかることになる．

著者らは，これらの従来研究に基づき 2 つのシステムを提案した[5]．1 つ目は図 2 に示すように仮想物体をカメラ映像中に合成し，仮想物体が家屋と重なるように手動で位置合わせすることで，家屋の位置を特定する．2 つ目は 2 視点の情報を用いて複数の家屋を同時に検出・トラッキングすることで，複数の家屋の位置を同時に特定する．どちらのシステムも，Visual SLAM によるカメラ位置姿勢推定を用いている．

本研究では，2 つ目のシステムに着目し，最新の物体検出器を利用するとともに，物体をトラッキングできている間の全てのフレームを用いることで，家屋の位置を精度よく推定する．

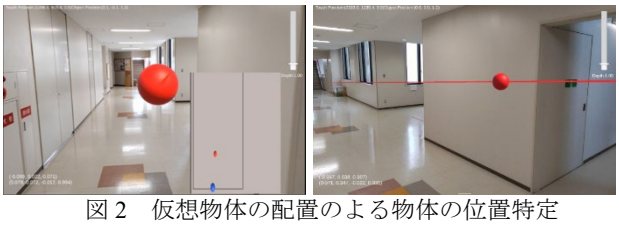


図 2　仮想物体の配置のよる物体の位置特定

3. 提案手法の概要

提案するシステムでは，モバイル端末のカメラ映像を用いて自己位置推定と環境地図作成を同時に行う Visual-SLAM（Simultaneous Localization and Mapping）[6]によって端末カメラの位置姿勢を算出する．そのうえで，YOLOv8[7]を用いた画像認識によって複数の家屋を対象物体として同時に検出・認識し，それらをフレーム間でトラッキングすることで，対象物体の 3 次元位置を特定する．以下では，提案したシステムについて順に説明する.

3.1 YOLOv8 による物体検出

本システムでは，モバイル端末単体で画像を認識してか

†大阪工業大学, Osaka Institute of Technology
‡防災科学技術研究所, National Research Institute for Earth Science and Disaster Resilience

ら物体の位置を特定するまで，高速に処理する必要がある．そこで，リアルタイム処理が可能である YOLOv8 を物体検出に用いる． YOLOv8 では，物体検出の結果として物体を囲むバウンディングボックス（以下，bbox）の情報，つまり中心座標(x, y)と幅・高さを出力する．また，一つの物体に対して複数の bbox が重複しないように，Non-Maximum Suppression の手法によって重なった領域を抑制する．

なお，学習データとして文献[9]で利用した画像に加えて，ミニチュアの家を撮影した画像を利用する．各画像で全壊している家屋としていない家屋にそれぞれ異なるラベルを付け，YOLOv8 を学習する．

3.2 物体の ID 割り当てとトラッキング

各フレームで物体を検出し，フレーム間で検出物体を対応付けることで，同一の物体としてトラッキングする．具体的には検出された物体の bbox を取得する．次に，前のフレームと現在のフレームの検出物体を局所性を考慮して対応付け，同物体には同じ ID を，異なる物体には新しい ID を割り当てる．ここでいう局所性とは，一つ前のフレームの bbox の中心座標と，現在のフレームの bbox の中心座標を比較した際に，カメラが大きく動かないことから物体の中心座標も大きく動かない性質を意味する．この考えに基づき，中心座標間の距離がある閾値以内の場合は同一物体，そうでない場合は異なる物体とみなす．

3.3 物体の 3 次元位置の特定

フレーム間でトラッキングされた物体の 2 次元座標および Visual-SLAM により推定した端末カメラの位置姿勢を用いることで，検出物体の 3 次元位置座標を推定する．図 3 に，初期フレームと端末を移動させた際のフレームから対象物体の座標を推定するときの座標の関係を示す．初期フレームおよび i 番目のフレームでの対象物体の bbox の中心座標をそれぞれ(x, y)，(x_i, y_i) とすると，座標(x, y)に対応する 3 次元点とそのときのカメラの光学中心を結ぶ直線および，もう一方のカメラ画像の座標(x_i, y_i) に対応する 3 次元点ともう一方のカメラの光学中心を結ぶ直線が得られる．この直線の交点が対象物体の位置$\boldsymbol{q} = (q_x, q_y, q_z)$として算出される．

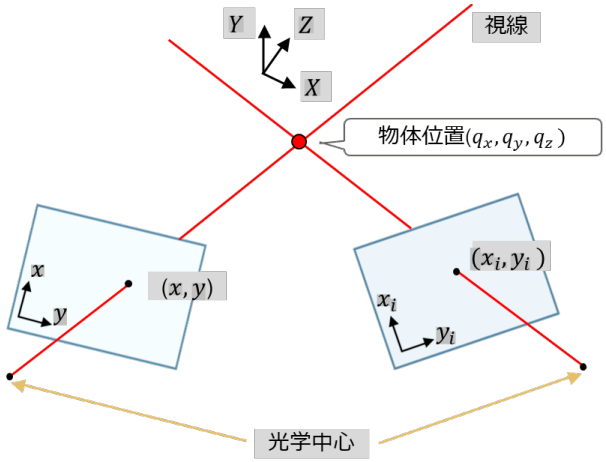


図 3　対象物体の座標推定

ただし，図4，5で示すように，現実には物体の画像上での抽出位置のずれや推定されるカメラ位置姿勢の誤差から，複数直線はねじれの位置になることが多く，複数直線が 3 次元空間中で交わることは稀である．そのため， 複数直線の最接近点を世界座標系として算出し，物体の 3 次元位置とする．

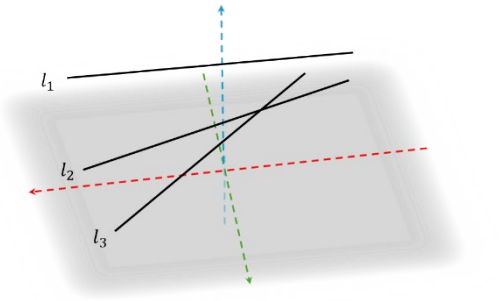


図 4　3 次元空間におけるねじれた複数直線

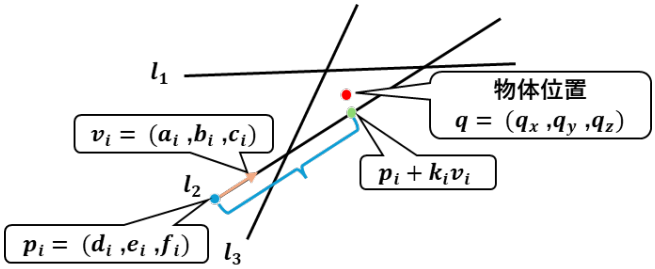


図 5　図 4 を上から見た複数直線

具体的には，図 5 のように i 番目のフレームでの物体検出時のカメラの位置の座標を$\boldsymbol{p_i} = (d_i, e_i, f_i)$，カメラの内部・外部パラメータとカメラ画像の座標から算出されるベクトルを$\boldsymbol{v_i} = (a_i, b_i, c_i)$，ただし，$\|\boldsymbol{v_i}\| = 1$とする．このとき，i 番目のフレームの直線上にある点は$\boldsymbol{p_i} + k_i\boldsymbol{v_i}$（ただし，$k_i$は係数）表され，直線上の点と点$\boldsymbol{q}$を結ぶ線分の距離の 2 乗和$E$は以下の式で表される．

$$E = \sum_i^n \|\boldsymbol{q} - (\boldsymbol{p_i} + k_i\boldsymbol{v_i})\|^2 \tag{1}$$

ここで$\frac{\partial \mathrm{E}}{\partial \mathbf{q}} = 0$，$\frac{\partial \mathrm{E}}{\partial k_i} = 0$（$\forall i$）を満たすとき，$E$が最小となる$\boldsymbol{q}$および$k_i$は以下のように算出される，

$$\boldsymbol{q} = \sum_i^n (\boldsymbol{p_i} + k_i\boldsymbol{v_i}) \tag{2}$$

$$k_i = \boldsymbol{q} \cdot \boldsymbol{v_i} - \boldsymbol{p_i} \cdot \boldsymbol{v_i} \tag{3}$$

式(2)(3)より複数直線の最近傍点$\boldsymbol{q}$は以下のように算出される．

$$\boldsymbol{q} = \boldsymbol{M}^{-1}\boldsymbol{w} \tag{4}$$

ただし，行列M，ベクトル$\boldsymbol{w}$は以下となる．

$$\boldsymbol{M} = \begin{bmatrix} \sum_i^n (1 - {a_i}^2) & \sum_i^n (-a_i b_i) & \sum_i^n (-a_i c_i) \\ \sum_i^n (-a_i b_i) & \sum_i^n (1 - {b_i}^2) & \sum_i^n (-a_i b_i) \\ \sum_i^n (-a_i c_i) & \sum_i^n (-b_i c_i) & \sum_i^n (1 - {c_i}^2) \end{bmatrix} \tag{5}$$

$$\boldsymbol{w} = \begin{bmatrix} \sum_i^n \left((1-{a_i}^2)d_i - a_i b_i e_i - a_i c_i f_i\right) \\ \sum_i^n \left(-a_i b_i d_i + (1-{b_i}^2)e_i - b_i c_i f_i\right) \\ \sum_i^n \left(-a_i c_i d_i - b_i c_i e_i + (1-{c_i}^2)f_i\right) \end{bmatrix} \quad (6)$$

3.4 システムのインタフェース

図 6 にシステムのインタフェースを示す．図の左上の Object Position は，端末の SLAMの取得位置から検出物体までの 3 次元座標を示している．図の左中央にある淡い黄色の領域は，算出された物体の 3 次元位置を俯瞰的に見たものを可視化したものであり，ミニマップとして表示している．このとき，ミニマップの下側がカメラの位置を示しており，オブジェクトのカメラからの相対位置を表現している．図の左下にある数値は，上から順にモバイル端末の位置を示す 3 次元座標，方向を示すクォータニオンである．図の中央付近にあるグレーで囲まれた領域はカメラ画像に対して画像処理を行った出力結果を表示している．また，図の右下にある Start ボタンを押すことによって処理を開始する．また，本システムに表示される位置の値は m 単位で出力される．

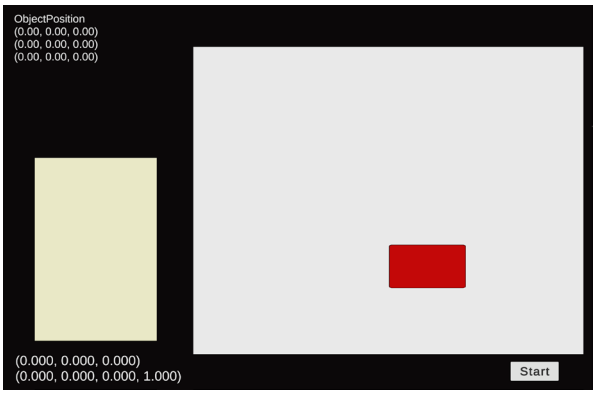


図 6　システムのインタフェース

4. 実験と考察

本実験では，YOLOv8 を用いて，モバイル端末で検出した物体をトラッキングし，検出物体の 3 次元位置が正しく算出されているかを検証した．システム開発には Unity 2021.3.33f1 を使用し，ARCore を利用することによって Android 用のアプリを作成した．また，モバイル端末としてタブレット PC（LAVIE Tab T1295/DAS）で検証を行った．また，本実験では，使用するモデルとして Pytorch モデルから TensorFlow Lite に変換したモデルを使用することで，高速化を行った．実験 1 では学習済みのモデルを用いて，実験 2，3 では，新たに生成した家屋検出のモデルを用いて検証を行った．以下では，3 種類の実験について述べる．

4.1 実験 1：YOLOv8 による物体検出

実験 1 では，家屋のモデルではなく，COCO データセットで学習済みの YOLOv8n モデル[7]を用いた．今回は，椅子を対象物体とし，3.0m と 4.2m 離れた場所に対象物体を配置し，検出した物体の 3 次元位置が正確であるかを検証した．図 7 に対象物体とカメラの移動方向の関係について示す．

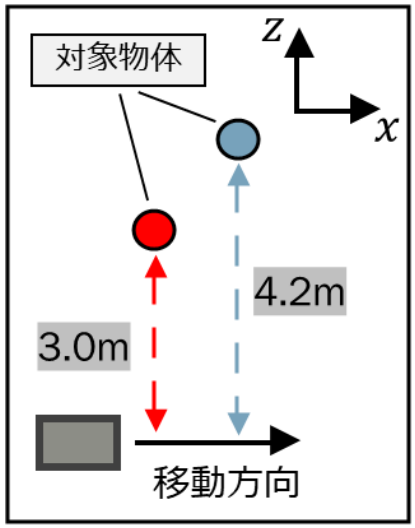


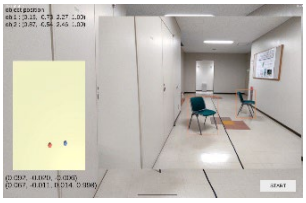
図 7　物体の対するカメラの移動方向（実験 1）

物体を検出した時の最初のフレームを図 8(a)，物体を検出してから 10 フレーム目を図 8(b)，数十フレーム目を図 8(c)に示す．また，それぞれのフレームでのおよび物体の位置を表 1 に示す．

表 1　フレームごとの物体の 3 次元位置（実験 1）

フレーム	3次元位置	
(a)	(0.00, -0.04, 0.06)	(-0.01, -0.01, -0.04)
(b)	(0.13, -0.73, 2.27)	(0.87, -0.54, 2.46)
(c)	(0.15, -0.91, 2.83)	(1.40, -0.88, 3.98)

(a)　1 フレーム目



(b) 10 フレーム目

(c) 数十フレーム目

図 8　対象物体の 3 次元位置

結果から，ObjectPosition が常に更新されていることから，物体のトラッキングができていることが確認できる．また，最初の 10 フレームではカメラの動きがまだ小さく，視差が大きくないため物体の位置が正しく算出されていないが，(c)では，真値に近い値が取得されており，物体が立体であることを考慮すると，おおむね正確な 3 次元位置が取得できていることがわかる．また，ミニマップから物体同士の位置関係も見てとれる．なお，作成システムはタブレット PC で約 5fps で動作した．

4.2　実験２：生成した家屋の学習モデルを用いて検証

実験 2 では，家屋を検出する YOLOv8 モデルを学習して用意し，検知した家屋の 3 次元位置が取得できているか検証した．実験 2 で用いた家屋のシーンを図 9 に示す．家屋のシーンでは，家屋のミニチュアを対象物体とし，実際のシーンの 1/150 のスケールであることを前提に実験を行った．また，それぞれ，30cm，55cm，40cm 離れた場所に対象物体を配置し，図 10 に検出した物体とカメラの移動方向の関係について示す．

本実験では，家屋の学習モデルの生成において，学習データ・検証データを生成するために，ラベリングツールとして labeling[10]を使用した．labeling では GUI で bbox とクラスラベルを使用することができる．学習モデルの生成には，家屋のミニチュアの精度を上げるため，文献[8]で使用されたラベリング済みの家屋のデータに新たにラベリングしたミニチュアの画像 12 枚を加えた計 169 枚用意し YOLOv8 モデルを学習した．学習に用いた YOLOv8 のモデルは YOLOv8n であり，エポック数 100 で学習モデルの生成後，タブレット PC 上で動作する TensorFlow Lite モデルに変換した．

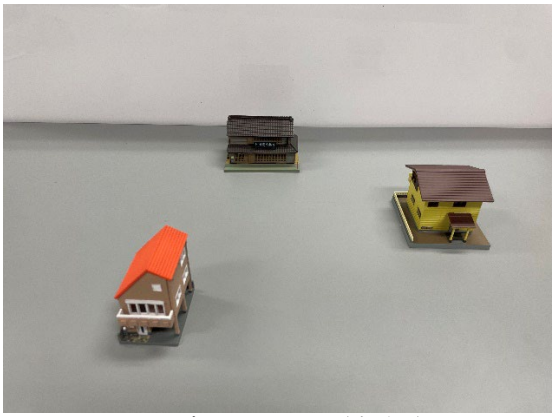


図 9　家屋のシーン（実験 2）

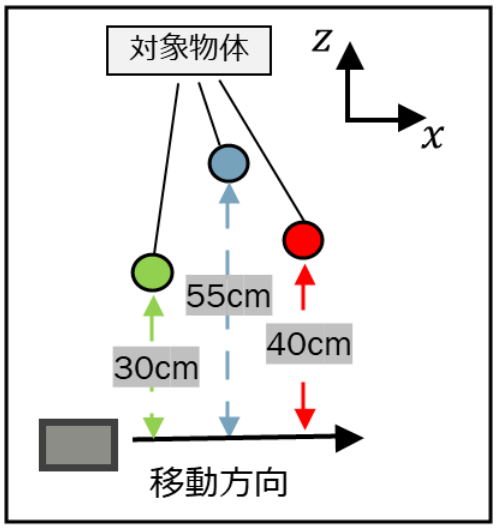
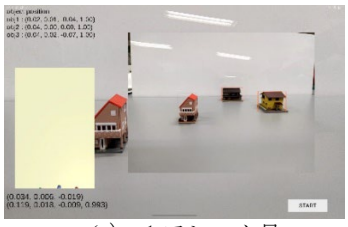


図 10　物体の対するカメラの移動方向（実験 2）

学習した家屋のモデルを用いた実験における，最初のフレームを図 11(a)，物体を検出してから 10 フレーム目を図 11(b)，数十フレーム目を図 11(c)に示す．また，それぞれのフレームでの物体の位置を表 2 に示す．

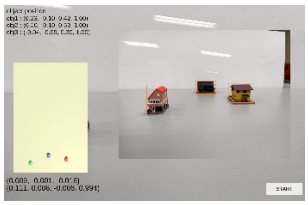
表 2　フレームごとの物体の 3 次元位置（実験 2）

フレーム	3次元位置		
(a)	(0.02, 0.01, -0.04)	(0.04, 0.00, 0.00)	(0.04, 0.02, -0.07)
(b)	(0.13, -0.05, 0.20)	(0.07, -0.04, 0.24)	(-0.02, -0.06, 0.24)
(c)	(0.23, -0.10, 0.42)	(0.10, -0.10, 0.53)	(-0.04 -0.09, 0.30)



(a)　1 フレーム目

(b) 10 フレーム目



(c) 数十フレーム目

図 11　対象物体の 3 次元位置と検知（実験 2）

結果から，家屋に bbox が描画されていることから，家屋のモデルが正しく生成できていることがわかる．また，実験 1 と同様に，ObjectPosition が常に更新されていることから，物体のトラッキングができていることが確認できた．最初の 10 フレームでは物体の位置が正しく算出されていないが，(c)では，真値に近い値が取得されており，物体は立体であることから，物体の奥行きを考慮すると，おおむね正確な 3 次元位置が取得できていることがわかる．また，実験 1 と同様，ミニマップから物体同士の位置関係も見てとれる．

ここでは，実際の 1/150 のスケールとしているため，実際のスケールを想定したとき，3 次元位置は表 3 のように表すことができる．数十フレームの間に約 23cm カメラが動いたことから，実際のスケールを想定したとき，開始位置から約 35 メートル移動すると 40～80m 離れた家屋のおおよその 3 次元位置が特定できることがわかる．

表 3　実際のスケールを想定した家屋の 3 次元位置

フレーム	3次元位置		
(a)	(3.0, 1.5, -6.0)	(6.0, 0.0, 0.0)	(6.0, 3.0, -10.5)
(b)	(19.5, -7.5, 30.0)	(-10.5, -6.0, 36.0)	(-3.0, -9.0, 36.0)
(c)	(34.5, -15.0, 63.0)	(-15.0, -15.0, 79.5)	(-6.00, -13.5, 45.0)

4.3　実験３：重なった家屋での検証

実験 2 では，カメラ画像に対して家屋同士の重なりを想定していなかったが，実験 3 では，家屋同士の重なりを考慮したときの家屋の検出と，検知した物体の 3 次元位置が正確であるかを検証した．実験 3 で用いた家屋のシーンを図 12 に示す．また，学習した家屋のモデルを用いた検出結果の，最初のフレームを図 13(a)，物体を検出してから数フレーム目を図 13(b)，数十フレーム目を図 13(c)に示す．

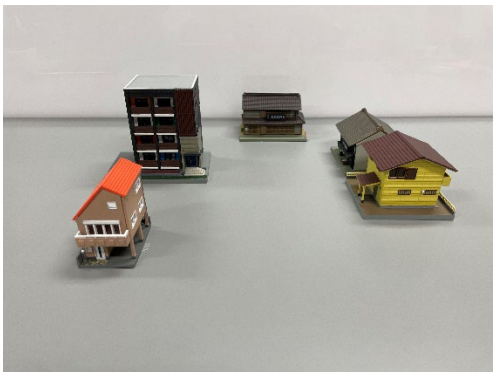
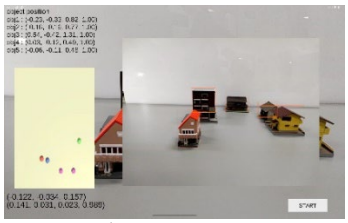
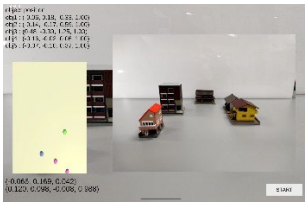


図 12　家屋のシーン（実験 3）



(a)　1 フレーム目

(b)数フレーム目



(c) 数十フレーム目

図 13　対象物体の 3 次元位置と検知（実験 3）

結果から，図 13(a)では，カメラ画像に写っているすべての家屋に対して bbox が描画されていることから，重なった家屋の場合に対しても検知していることがわかる．しかし，同図(b)のように，タブレット端末を動かしたとき，家屋が検知さていないことがわかる．また，同図(c)では，すべての家屋に対して，bbox が描画されていないことがわかる．

家屋が検知できなかった原因としては，角度やカメラの移動により家屋が正しく検出されていないことから物体の 3 次元位置が正しく算出されなかった．また，家屋の学習用画像が不十分であったと考えられる．

(c)において，bbox が描画されていない原因としては，本実験では，現在のフレームと 1 フレーム前の 2 フレーム間で物体のトラッキング処理を施しているため，検知した物体が 1 度でもトラッキングから外れると，再び同じ物体を認識したときに新しい物体と判断し，新たな ID を割り当てている．また，実装上 500 個の物体を上限として ID を割り当てているため，501 個以上の物体が検出された場合，検出の対象外となってしまうため bbox が描画されなかったのだと考えられる．

5. まとめ

本研究では，モバイル端末からカメラ画像を取得し，画像認識によって複数物体を同時に検出・認識し，それらをフレーム間でトラッキングし，Visual-SLAM を用いることで物体の 3 次元位置を特定する手法を提案した．COCO で学習済みのモデルを用いた実験 1 では，検出した物体をトラッキングし，3 次元位置を取得することが確認できた．実験 2 では，学習した家屋のモデルを生成し導入することで，家屋を検知し，実験 1 と同様に検出した物体をトラッキングし，3 次元位置を取得することが確認できた．実験 3 では，重なった家屋のシーンで実験を行ったが，カメラの移動や角度によって検知できない家屋がみられ，数十フレーム後では bbox が描画されなかった．これには，家屋の学習用画像が不十分であり，トラッキングによる ID 割り当てが原因であると考えられる．

今後の課題として，家屋の学習画像を増やすことで，検出の精度の向上や新たなトラッキングの手法を提案することがあげられる．また GNSS や電子コンパスなどにより地球上での絶対的な位置を取得するシステムの開発を検討する．

謝辞 本研究は，JSPS 科研費 23K23017 の助成を受けて実施した．

文　献

[1] 伊勢　正，古森　和城，磯野　猛，臼田　裕一郎：“MR デバイスを用いた被災状況把握ツールの基礎研究”，防災科学技術研究資料 第 478 号，pp.1-16, 2022.

[2] 織田　和夫：“SfM の概要とバンドル調整”，写真測量とリモートセンシング資料 第 55 号，pp.206-209, 2016.

[3] 画像センサ - 改善・活用事例集：https://www.fa.omron.co.jp/product/applications/sensors/vision-sensors_machine-vision-systems/,（閲覧日 2024 年 7 月 25 日）

[4] Jarkko Polvi, Takafumi Taketomi, Goshiro Yamamoto, Arindam Dey, Christian Sandor, Hirokazu Kato: “SlidAR: A 3D positioning method for SLAM-based handheld augmented reality”, Computers & Grahics, Vol. 55, pp.33-43, 2016.

[5] 岸川　雄星，北野　太一，河合　紀彦，鈴木　基之，伊勢　正：“遠方からの被災家屋の位置特定に関する検討”，電子情報通信学会 技術研究報告, MVE2023-50, 2024.

[6] ARCore とサポートされている開発環境の概要：https://developers.google.com/ar/develop?hl=ja,（閲覧日 2023 年 10 月 22 日）

[7] Glenn Jocher，Ayush Chaurasia，Jing Qiu: “Ultralytics YOLOv8”, 2023.

[8] 木村 裕貴, 木内 一隆, 河合 紀彦, 鈴木 基之, 伊勢 正: “YOLO による物体検出を用いた全壊した住家の検出”, 情報処理学会第 85 回全国大会講演論文集, 6R-06, 2023.

[9] Tzutalin: “LabelImg. Git code (2015)”，https://github.com/tzutalin/labelImg,（閲覧日 2024 年 7 月 26 日）