開発レポート　Development Report

物体検知・自作学習アプリ「M-TIC Detect」の開発

― COCO 学習済モデルと自前写真による追加学習 ―

M-TIC Detect: Object Detection with a COCO-Pretrained Model and On-Device Learning from User Photos

レポート番号	MTIC-DR-2026-005
団体	NPO 法人松戸技術革新センター M-TIC　松戸ロボット研究会
著者	大城 公平
発行日	2026 年 10 月 25 日
対象	M-TIC Detect v0.1.3（ソフトウェア）
分類	開発レポート／物体検知・機械学習ソフト
機体状態の基準日	2026 年 9 月 21 日

本レポートの数値と機体状態について　本検証系は研究開発段階にあり随時変更されるため、記載の仕様は基準日時点のものであり、現在の実機と一致することを保証しない。数値には種別を付す：**公称**＝カタログ・設計値、**実測**＝実際に計測した値（記載桁が測定分解能）、**概算**＝目測・推定値。

概要

ロボットが周囲の物体を「何であるか」まで識別できれば、自動運転（M-TIC AutoDrive）の回避判断や作業ロボットの対象認識に使える。「M-TIC Detect」（正式名称：松戸技術革新センター 物体検知・学習アプリケーション）は、COCO 80 種を学習済みの SSD-MobileNetV2（TFLite）で人・ペットボトル・空き缶などを枠と確信度[%]で表示し、さらに GUI から自分で写真を撮って「これは○○」と教える追加学習（MobileNet 特徴量＋kNN）を備えたアプリである。特に、ロボット自体が「人」と誤認識される問題に対し、撮影した写真でロボットをロボットとして再判定できることを狙った。Raspberry Pi 4B で動く軽さと、他プログラムと合体できる余裕を優先し、結果は UDP／JSONL で出力する。PC 上の試験では、640×480 の映像を毎秒 17〜19 フレーム（推論 1 回 10〜19 ミリ秒）で処理し、人やスマートフォンを検知できた。学習済みモデルに含まれない「ねじ」は、学習前は検知されなかったが、写真 25 枚で追加学習させると、確信度 95%で「ねじ」と判定されることを確認した。一方で、掲示板を冷蔵庫と誤検知する例もあった。この結果は事務所の机上で、学習時と近い条件で得たものである。ロボットの再判定と Raspberry Pi 上の処理速度はまだ確かめておらず、次段階で計測してデータをもとに評価したい。

1. 背景と目的

1.1 開発背景

カメラ認識アプリ Vision Studio（MTIC-DR-2026-004）は色や形に基づく認識であり、物体の種類は識別しない。自動運転や作業ロボットでは「人か、物か、自分たちのロボットか」を区別する必要がある。特に、既存の学習済みモデルでは M-TIC のロボットが「人」と認識される問題があった。

1.2 目的と目標

学習済みモデルで一般的な物（COCO の 80 種類）を検知できる状態を初期状態として持ち、さらに GUI から自分で写真を撮って追加学習できるアプリを、プログラムが得意でない人でも操作できる形で作る。

目標：COCO 80 種の検出と、自前写真による追加学習（例：ロボットをロボットとして認識）が GUI だけで行えること。

2. システム構成

2.1 開発環境と基本的な作り方

Python で書き、Thonny で開発した。推論は TFLite、カメラと画像処理は OpenCV、画面は Tkinter という組み合わせで、Raspberry Pi 4B でも動く軽さを保っている。検知結果は UDP と JSONL ログで出すので、AutoDrive など別のプログラムから受け取れる。物体検知には、学習済みの軽量な深層学習モデルを TFLite 形式で用いる（図 1・図 2）。



図 1　開発環境と基本構成



図 2　M-TIC Detect の処理の流れ

※ 本レポートでは処理の考え方と構成のみを示し、ソースコードおよび調整値は掲載しない。

2.2 構成

項目	内容
正式名称	松戸技術革新センター 物体検知・学習アプリケーション（略称 M-TIC Detect）
版	v0.1.3
実行環境	Python（Thonny）。開発は Windows、実行は Raspberry Pi 4B。推論ライブラリは OS ごとに TFLite 互換のものを使い分け
検知モデル	SSD-MobileNetV2（TFLite、COCO 80 クラス学習済）。枠＋クラス名＋確信度[%]を表示
追加学習	MobileNet 特徴量抽出＋kNN 分類。GUI から写真を撮り、ラベルを付けて登録。COCO の枠に対して再判定を行う
カメラ	USB カメラ。カメラ処理は M-TIC のカメラ系 4 アプリで共通化
出力	UDP／JSONL
学習の手順（GUI）	① 覚えさせたい物の名前を入力 ②写真を切り取る範囲を選ぶ（最も大きく検知された枠／画面中央の枠／画面全体）③ 1 枚ずつ、または連続撮影（0.25 秒間隔）で撮る ④学習して保存（次回起動時も使用）

Copyright © 2026 Matsudo Technology Innovation Center. All rights reserved.

項目	内容
判定の範囲	学習した物（自作クラス）は、COCO モデルが出した枠の再判定と、画面中央の枠で判定する（画面全体を細かく探す設定もあるが処理が重い）
版の履歴	v0.1.1：日本語・全角スペース入りフォルダでもモデルが開けるよう修正。v0.1.2：共通カメラモジュール導入。v0.1.3：日本語のフォルダ名・クラス名で写真が保存されない不具合を修正

検知と追加学習の関係を図 3 に示す。学習済みモデルが出した枠に対して、枠の中の特徴を自分で登録した写真と比べ、近ければ登録した名前に置き換える。モデルそのものを作り直さないため、写真を撮って名前を付けるだけで、その場で反映できる。

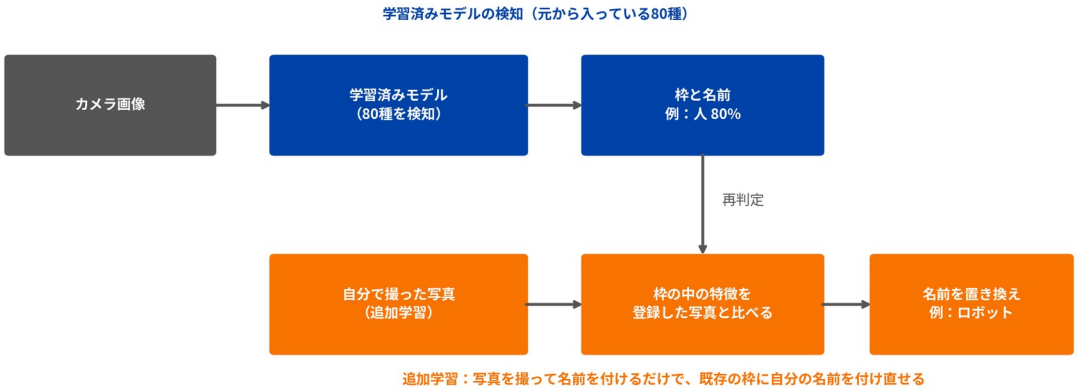


図 3　学習済みモデルの検知と追加学習の関係（概念図）

2.3 設計方針

- ラズパイ 4B で動く軽さを優先し、YOLOv5 ではなく SSD-MobileNetV2 を選択
- 追加学習はモデルの再訓練ではなく、特徴量＋kNN の軽量方式とし、ラズパイ上でも写真を追加するだけで即反映できる構成にした
- 他プログラム（AutoDrive 等）と合体できるよう、検知部を分離し出力を UDP／JSONL に統一
- カメラ処理の共通化は本アプリで先行検証し、他のカメラ系アプリへ順次展開

3. 検証方法と結果

3.1 検証方法

Windows PC に USB カメラを接続し、640×480 の解像度で事務所内の人・物を映して学習済みモデル（COCO）の検知を確認した。次に、COCO に含まれない「ねじ」を映して学習前に検知されないことを確認したうえで、画面中央の枠にねじを置き、連続撮影などで角度や位置を変えながら合計 25 枚の写真を登録して追加学習させ、学習後の検知を確認した（図 6）。

Copyright © 2026 Matsudo Technology Innovation Center. All rights reserved.

3.2 結果

項目	結果
処理速度（PC、640×480）	毎秒 16.7〜18.9 フレーム、推論 1 回 10〜19 ms（実測、画面表示）
COCO：人	確信度 80 %で検知（図 4）
COCO：スマートフォン	確信度 87 %で検知（図 5）
COCO：誤検知の例	壁の掲示板を「冷蔵庫」と確信度 55 %で誤検知（図 4 左端）
追加学習：ねじ（学習前）	COCO に含まれないため検知されない（図 7 左）
追加学習：ねじ（学習後）	写真 25 枚で学習後、確信度 95 %で「ねじ」（自作クラス）と判定（図 7 右）。今回の試験条件（机上・同じカメラと照明）での結果

ロボットをロボットとして再判定できるかと、Raspberry Pi 4B での処理速度は、まだ試せていない（5.1 節）。

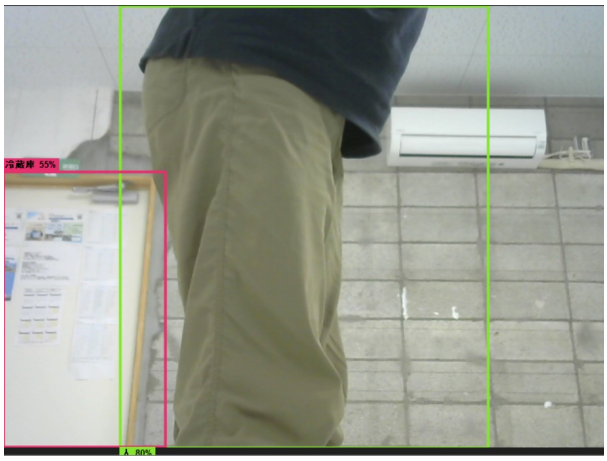


図 4　学習済みモデルによる検知（人 80 %。左端の掲示板を冷蔵庫と誤検知）



図 5　スマートフォンの検知（87 %）

Copyright © 2026 Matsudo Technology Innovation Center. All rights reserved.



図 6　追加学習の画面（画面中央の枠に「ねじ」を置いて連続撮影）

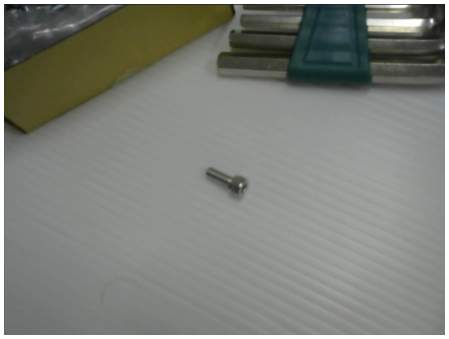


図 7 左　学習前：ねじは検知されない

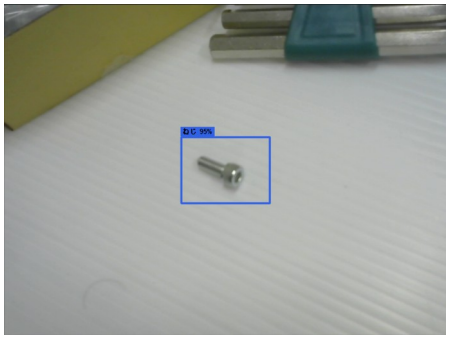


図 7 右　学習後（写真 25 枚）：ねじ 95 %

※ 画面は PC 上で取得。調整値の表示部分は掲載していない。

4. 課題と考察

- **kNN 追加学習の限界：**特徴量＋kNN は少ない写真で即反映できる反面、背景や照明が変わると精度が落ちやすい。何枚撮れば安定するかはまだ確かめておらず、今後、枚数を変えて比べたい。
- **判定範囲の制約：**学習した物は、COCO モデルが出した枠と画面中央の枠で判定する設計のため、COCO が検知しない物が画面の端にある場合は検出できない可能性がある（今後確かめたい）。画面全体を細かく探す設定は処理が重く、Raspberry Pi では使いにくい。
- **学習済みモデルの誤検知：**掲示板を冷蔵庫と判定したように（図 4）、COCO モデルは形の似た物を別の物と誤ることがある。確信度のしきい値を上げれば誤検知は減るが、見逃しが増える。
- **実機での確認：**ラズパイ 4B での処理速度はまだ計測していない。SSD-MobileNetV2 でも fps が不足する場合は入力解像度の削減が必要。
- **Windows／ラズパイの実行環境差：**Windows 用と Raspberry Pi 用の推論ライブラリの差による挙動の違いは未確認。

Copyright © 2026 Matsudo Technology Innovation Center. All rights reserved.

5. 今後の展望

5.1 検証計画

計測項目	方法	判定基準
ラズパイ 4B の処理速度	画面表示から fps を記録（入力解像度 2 種）	AutoDrive と同時実行できる fps
ロボットの再判定	ロボットを 10・30・50 枚撮影して登録し、別角度 20 枚で正答率を記録	正答率と必要枚数を記録
COCO の検出率	人・ペットボトル・空き缶を各 20 回提示し検出回数を記録	検出率を記録し基準を設定

5.2 応用の可能性

AutoDrive の障害物判定と組み合わせ、「人なら停止、物なら回避」のように種類で挙動を変える構成へ発展させる。このあとは、学習させる物・写真の枚数・撮影環境を変えながら、少ない写真でも安定して判定できる条件を探る。

5.3 カメラ認識技術の統合（構想）

M-TIC で開発してきたカメラ認識ソフト（床面認識・ライン認識・物体検知・色や形の認識）を組み合わせて 1 つの自動運転に使う構想は、CWM_CamLine のレポート（MTIC-DR-2026-006）の 5.3 節にまとめている。Detect は、その中で「それが何か」を見分けて避け方を変える役を担う。

6. まとめ

COCO 学習済モデルによる物体検知と、GUI からの写真撮影による軽量な追加学習を備えた M-TIC Detect v 0.1.3 を開発した。PC 上の試験で、人・スマートフォンの検知と、学習前は検知されなかった「ねじ」が、写真 25 枚の追加学習後に確信度 95%で判定されることを確認した。背景や照明が変わった場合や Raspberry Pi 上での挙動は、4 章で述べたとおり別途確かめる必要がある。ロボットの再判定と Raspberry Pi 上の処理速度を次段階で計測し、AutoDrive との統合に進む。

参考資料

TensorFlow, “SSD MobileNet V2 (COCO)” モデルカード

Google, “LiteRT” ドキュメント

T. Cover, P. Hart, “Nearest neighbor pattern classification”, IEEE Trans. Inf. Theory, 1967

改訂履歴　初版（2026-09-30）　2026-10-25　文章を見直し、誤記を訂正。

Copyright © 2026 Matsudo Technology Innovation Center. All rights reserved.