開発レポート　Development Report

カメラ自動運転ソフト「M-TIC AutoDrive」の開発

― カメラ１台による床認識・障害物回避・マーカー追従走行 ―

M-TIC AutoDrive: Monocular-Camera Floor Recognition, Obstacle Avoidance and Marker Following

レポート番号	MTIC-DR-2026-003
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 AutoDrive v0.3.1（ソフトウェア）
分類	開発レポート／自動運転・画像認識ソフト
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

ライン・磁気テープなどの誘導設備がなく、LiDAR 等の距離センサも持たない機体が、USB カメラ 1 台の映像だけで床（走行可能範囲）を認識し、障害物を避け、目標マーカーへ向かって走るための制御ソフト「M-TIC AutoDrive」（正式名称：松戸技術革新センター 汎用自動運転・走行技術研究開発アプリケーション）を開発した。Raspberry Pi 4B でオフライン動作することを最優先に、床の HSV 色モデル＋エッジ検出による走行可能範囲の推定、画面を 9 セクタに分けた空き判定、ArUco 4×4 マーカーおよびオレンジ蛍光マーカーの追従、廊下中央追従を実装し、自動運転バス風のダッシュボード UI で進路・左右モーター出力・速度を表示する。制御指令は UDP で別プロセスの機体制御ソフトへ渡す構成とし、既存の機体制御ソフトは改変しない。PC 上の試験では毎秒 78〜83 フレームで処理でき、床の色を学習させると、灰色の床と灰色のブロック壁の境界をカメラ映像だけで概ね正しく検出し、オレンジ色のマーカーへ向かう走行指令を出力できた。一方、床と壁が完全に同じ色・質感の場合は誤認識の可能性が残る。実機での走行成功率と障害物検出距離はまだ計測しておらず、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

M-TIC が研究開発しているピッキングロボット CWM と運搬ロボット CBV は Raspberry Pi 4B を搭載し、ライントレースや手動操作で走行している。物が多く散らかった事務所のような場所では床にラインを引けず、LiDAR（レーザーで周囲の距離を測るセンサー）や深度カメラは高価なため、現時点では導入していない。カメラは既に搭載しており、カメラだけで走れれば追加ハードなしで自律走行に踏み出せる。

1.2 目的と目標

USB カメラ 1 台の映像から床と壁・物を区別して走行可能範囲を推定し、障害物を避けながら目標マーカーへ向かう制御ソフトを、完全オフライン・Raspberry Pi 4B で動く軽さで実装する。

目標：廊下をカメラだけで中央追従走行できること。広い場所ではオレンジ色の蛍光マーカーを追って走れること。

2. システム構成

2.1 開発環境と基本的な作り方

本アプリは Python で記述し、開発環境には Thonny を用いた。Thonny は初学者向けに設計された統合開発環境で、Windows 上で作成したプログラムを多くの場合そのまま Raspberry Pi 4B でも実行できる（使うライブラリによっては OS ごとの違いに注意が必要）。カメラ映像の取得と画像処理には標準的な画像処理ライブラリ OpenCV を用い、画面（GUI）は Python 標準の Tkinter で構成した。処理結果は UDP（同じネットワーク内のプログラム同士で短いデータを送り合う通信方式）で機体制御ソフトへ渡す。これにより、画像認識と機体制御を別々のプログラムとして開発・更新できる（図 1・図 2）。



図 1　開発環境と基本構成



図 2　M-TIC AutoDrive 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 ソフトウェア構成

項目	内容
正式名称	松戸技術革新センター 汎用自動運転・走行技術研究開発アプリケーション（略称 M-TIC AutoDrive）
版	v0.3.1（2026-09-21）
実行環境	Python（Thonny）。Windows で開発、Raspberry Pi 4B で実行。完全オフライン（Wi-Fi 不要）
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
床認識	床の HSV 色モデル（3 種から選択）＋エッジ検出で走行可能範囲を推定。壁と床を区別
障害物判定	画面を 9 セクタに分割し、各セクタの空き（床率）を判定
目標追従	ArUco 4×4 マーカー／オレンジ蛍光マーカーの追従、廊下中央追従
出力	UDP で左右モーター出力・状態を機体制御ソフトへ送信（機体制御ソフトは改変しない）

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

項目	内容
UI	自動運転バス風ダッシュボード：走行進路・走行可能範囲・左右モーター出力・Accel/Brake/Steering/Speed/Distance/Time を表示。カメラ開始/停止、カメラ選択プルダウン、バックエンド名と処理時間[ms]表示

2.3 床の学習と 9 セクタ判定の考え方

走れる範囲の推定は 2 段階で行う（図 3）。まず、画面下部の枠に映った床の色を「走れる床」として学習し、学習した色に近い部分を走行可能範囲として塗り分ける。色が変わる境目を、走行可能範囲と走行不可範囲の境界とみなす。次に、画面を縦横 3 分割した 9 つの区画（セクタ）ごとに床の割合を求め、割合が高い方向を進路とし、割合が低い区画を障害物や壁として避ける。

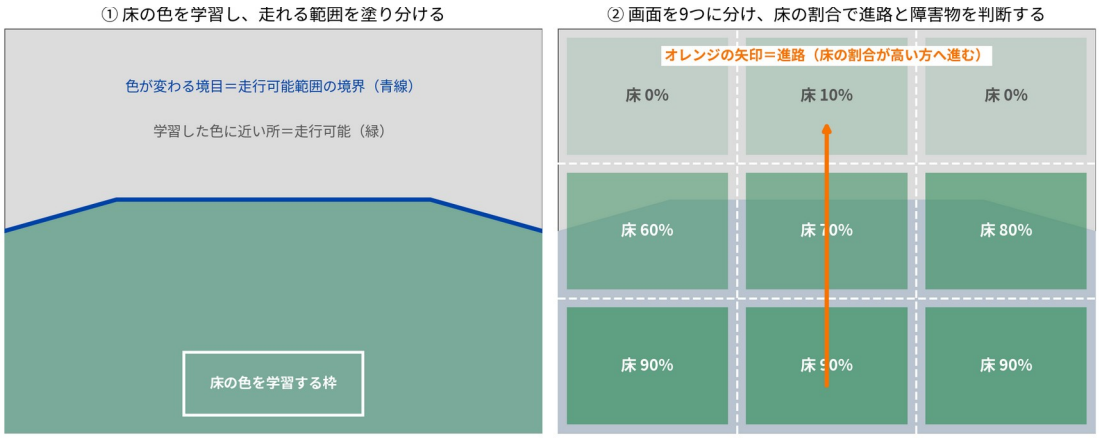


図 3　床の学習と 9 セクタ判定の考え方（概念図）

2.4 版の履歴

版	変更
v0.1.1	GUI パネルの押し出し表示を修正
v0.1.2	カメラ選択プルダウンを追加
v0.2.0	Raspberry Pi 向け軽量化、USB カメラ 0 番を既定に
v0.3.0	自動運転バス風ダッシュボード、壁／物判別、廊下中央追従、オレンジ蛍光マーカー追従、床モデル 3 種
v0.3.1	共通カメラモジュール導入（カメラ切替の高速化、開始／停止）

2.5 設計方針

- Raspberry Pi 4B で動くことが最優先。Windows 側の高速化のためにラズパイ側の動作を変えない
- 追加のセンサーを使わず、有名で軽量な手法（色モデル＋エッジ、ArUco）でカメラ 1 台でできる範囲を見極める。距離センサーとの連携は今後の展望（5.2 節）とする
- 機体制御ソフトとは別プロセスとし、UDP で疎結合にする（Vision Studio と同方針）

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、事務所の床（灰色）とコンクリートブロックの壁（灰色）がある場所で、床の上に円錐形のマーカーを置いて映した。まず床の色を学習させる前と後で走行可能範囲の表示

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

を比較し、次にオレンジマーカー追従を有効にして走行指令の出力を確認した。床の学習は、画面下部の枠内に映った床の色を「走れる床」として登録する機能である。カメラは手で固定し、機体には搭載していない。

3.2 結果

項目	結果
処理速度（PC）	毎秒 78〜83 フレーム（実測、画面表示）
床を学習する前	走行可能範囲が表示されず、前方が空いていないと判定して停止（図 4 左）
床を学習した後	走行可能範囲を緑、走行可能範囲と走行不可範囲の境界を青線で表示。灰色の床と灰色のブロック壁の境目を境界として概ね正しく検出（図 4 右）
マーカーの扱い	床の上の円錐マーカーを障害物として床から除外し、オレンジマーカーを目標として検出
オレンジマーカー追従	目標を検出して「オレンジマーカーへ走行」の状態に移り、左右同出力（各 21 %）の走行指令を出力（図 5）

Raspberry Pi 4B での処理速度、実機での走行成功率、障害物の検出距離と見逃し率は、今回は測れていない。5.1 節の計画に入れてある。

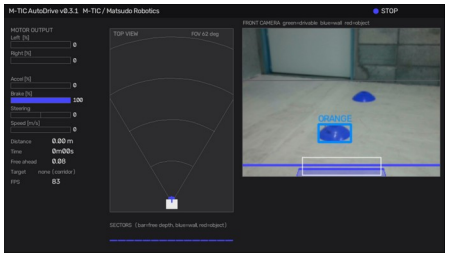


図 4 左　床を学習する前

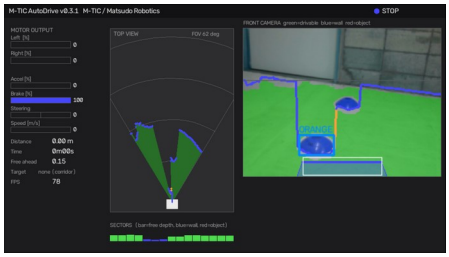


図 4 右　床を学習した後（緑＝走行可能範囲、青線＝走行可能範囲と走行不可範囲の境界）

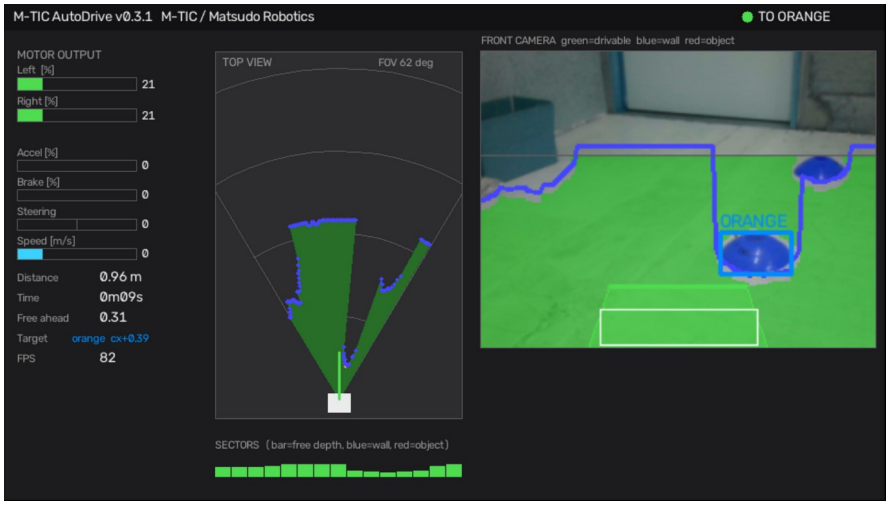


図 5　オレンジマーカーへ向かう状態（上面図に走行可能範囲、右にカメラ映像）

※ 画面は PC 上で取得。調整値の表示部分は掲載していない。カメラ映像の色は表示上の都合で実際と異なる場合がある（本ソフトは表示の際に色を反転させる処理を行っており、オレンジ色のマーカーは青く表示される）。

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

4. 課題と考察

4.1 カメラ（RGB）だけで走る自動運転の限界

本ソフトは床の「色」とエッジを手がかりに走行可能範囲を推定する。この方式は追加のセンサーが不要で軽い反面、原理上、以下の状況で誤認識が起こりうる。

状況	誤認識が起こる理由
床と壁・物が同じ色	色で床を判定するため、床と色が近い壁や物体も「走れる床」と判定してしまう
光の反射・影	照明の映り込みや影で、同じ床でも色が違って見える
床の模様・つなぎ目	模様やタイルの境目を、障害物や床の端と判定しうる
距離の把握	カメラ１台（単眼）では、障害物までの実際の距離を直接測れない
透明な物体	ガラスや透明なケースは色の違いとして現れにくい
カメラの死角	画角の外や、車体の真下に近い低い障害物は見えない

今回の試験環境では、床と壁がどちらも灰色であるにもかかわらず、床の色を学習させた後は、床と壁の境目を走行可能範囲と走行不可範囲の境界として概ね正しく検出できた（図４右・図５）。床と壁のわずかな明るさ・色合いの違いと、境界に現れるエッジが手がかりになったと考えられる。ただし、床と壁が完全に同じ色・同じ質感の場合や、照明で両者の見え方がそろってしまう場合については検証しておらず、上表の誤認識が起こる可能性は残る。

図６は床に置いた段ボール箱を映した例である。箱は壁・障害物として床から除外され、正しく「走れない領域」と判定された。一方、画面左側の床の一部は走行可能（緑）と判定されず、床であるにもかかわらず走れない領域として扱われた。照明による明るさのむらが原因と考えられる。この誤りは「走れる所を走れないと判断する」安全側の誤りであり衝突にはつながらないが、走行できる範囲が狭くなる。逆に、壁や物を床と判断する危険側の誤りは今回の試験では観察されなかったが、床と壁が完全に同色の場合には起こる可能性がある。

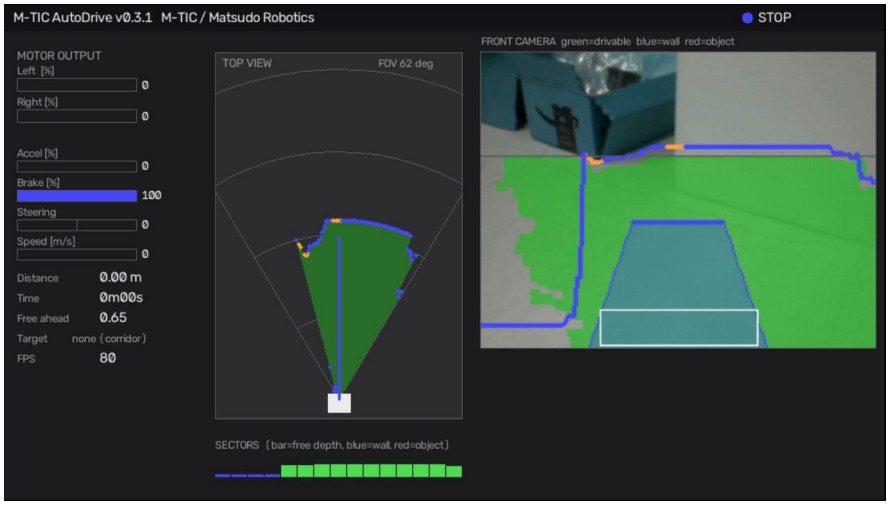


図６　段ボール箱の前での認識（箱は除外されたが、左側の床の一部を走行可能と判定できていない）

このため本ソフトは、床と壁・障害物の色がはっきり異なる環境での走行を前提とする。色だけに頼らず距離の情報を加えることが、実用化に向けた最も重要な改善点である。

4.2 本結果の解釈上の注意

- 本結果は、事務所の特定の床・壁・照明の条件で、カメラを手で固定して得たものであり、他の環境で同じ結果を保証するものではない。

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

- 処理速度は Windows PC での値であり、Raspberry Pi 4B では低下する可能性がある。
- 走行指令は出力を確認したのみで、機体を実際に走らせた結果ではない。画面の走行距離・時間の表示はソフト内の推定値であり、本レポートの結果には用いていない。

4.3 その他の課題

- **実機での確認：**v0.3.1 は PC 上の確認にとどまる。ラズパイ 4B での処理速度と、実走行時の床モデルの頑健性（照明変化・床の模様・反射）が主な不確定要素である。
- **床認識の限界：**HSV 色モデルは床色が一様でない場所や、床と同系色の物体に弱い。壁と床の区別はエッジに依存するため、境界が不明瞭な場所では誤判定が起きうる。
- **マーカー依存：**広い場所ではマーカーが視野に入っていることが前提。マーカーを見失ったときの探索動作は未実装。
- **共通モジュールの制約：**カメラ処理は 4 アプリで共通化しているため単独では変えられず、変更時は全アプリへの影響を確認する。

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	画面の ms 表示から fps を算出、各床モデルで記録	走行制御に十分な fps を記録し基準を設定
廊下中央追従	事務所の廊下で 10 回走行、壁接触の有無を記録	接触なしの回数／10
障害物回避	廊下に段ボール等を置き、停止・回避の可否と距離を記録	接触なし・停止距離を記録
マーカー追従	距離を変えて検出可能距離を記録	最大検出距離［m］
誤認識の条件	床と壁の色の組み合わせ・照明を変えて走行可能範囲の表示を記録	誤認識が起きる条件を一覧化
距離センサー導入後の比較（導入時）	深度カメラまたは LiDAR を導入後、同じコースでカメラのみの場合と比較	誤認識の件数が減ること

5.2 応用の可能性と距離センサーとの連携

4.1 節の限界を補うには、距離を直接測れるセンサーとの連携が有効である。候補は、色の画像と同時に画素ごとの距離を取得できる深度カメラ（RGB-D カメラ）と、レーザーで周囲の距離を測る LiDAR である。いずれも現時点では高価なため導入していないが、導入できれば、色による床の推定に距離の数値を組み合わせることで、床と壁が同じ色でも「近くにある面＝壁・障害物」と区別できる可能性があり、障害物までの距離に応じた減速・停止も期待できる。

CWM・CBV への搭載を想定している。物体検知アプリ（M-TIC Detect、MTIC-DR-2026-005）と組み合わせ、「何が」障害物かを識別して回避方針を変える構成へ発展させる。

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。AutoDrive はその土台として、走れる範囲を決める役を担う。

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

6. まとめ

USB カメラ 1 台で床認識・障害物判定・マーカー追従を行う M-TIC AutoDrive v0.3.1 を開発し、PC 上で、床の色の学習、走行可能範囲の表示、障害物の除外、オレンジマーカーへ向かう動作を確認した。床の色を学習させることで、灰色の床と灰色の壁の境界をカメラ映像だけで概ね検出できたことは、今回いちばん確かめたかったことである。色に頼る方式のため、床と壁が同じ色の環境や光の反射がある環境では誤認識が起こりうることを整理した。今後はラズパイ 4B での処理速度と実機走行の成功率を計測するとともに、将来、深度カメラや LiDAR を導入できた段階で距離情報と連携し、限界の克服を図る。

参考資料

OpenCV, “Detection of ArUco Markers” ドキュメント

Python Software Foundation, “tkinter — Python interface to Tcl/Tk”

Thonny, Python IDE for beginners（thonny.org）

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.