発泡プラスチック製品の離型不良検出システムのプロトタイプの開発

Development of a prototype system for detecting mold release defects in foam plastic products

電子・機械技術部　電子・情報科　山田昌幸　鎌田直樹

発泡プラスチック製品の離型不良発見を省力化するため、Python 及び OpenCV を使用して、カメラから取得した映像から不良の発生を判定するシステムを試作した。システムはミニ PC と撮影用カメラ、画像処理用プログラムで構築した。このシステムでは、動作している金型をカメラで撮影し続け、離型不良が発生したときに画面に表示し、離型不良発生時のカメラ画像を保存するようにした。これを現場へ試験導入し、現場での離型不良発見の省力化の検証作業を行った。

Key words: 画像処理、自動判別、OpenCV、Python、GUI アプリケーション

１．緒言

応募企業では、様々な形状の発泡プラスチック製品を製造している。発泡プラスチック製品を作る際、凸型と凹型の金型で成形し、固まった後に金型を開き、下にあるラインに製品を落下させる（離型）。この時まれに金型から製品が離れずそのままになってしまうことがある（離型不良）。加工機側にも金型が閉じたときの圧力から離型不良を検知するシステムがあるが、検出率が芳しくない。

そのため作業員がたびたび金型の横から目視で確認していたが、金型の場所が作業員の作業する場所から少し離れており、途中に階段もあるため、時間がとられるうえ転倒の危険性もあり、かつ見逃しなども発生していた。そこで、画像処理技術等により離型不良判定を行い、不良発見を省力化するシステムを構築したいという要望があった。

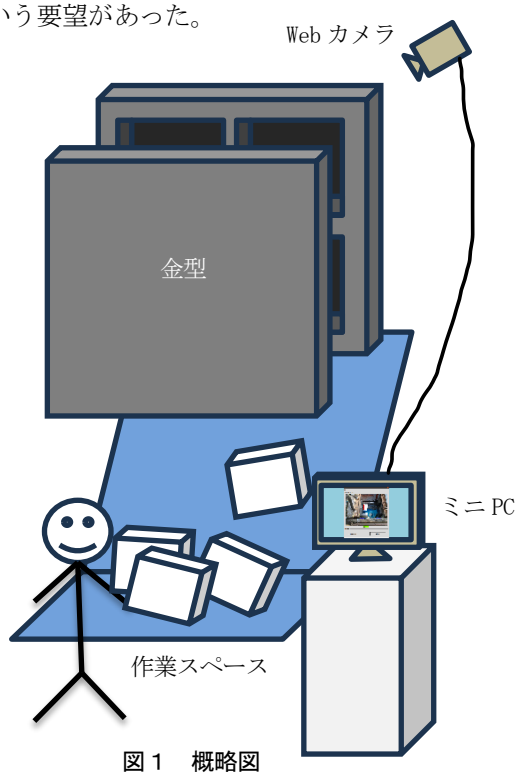

図１　概略図

しかし当該企業には画像処理やシステム構築に関するノウハウが無く、自力で開発するにはプログラミングの習熟が必要であり、実際に構築・導入に至るには時間がかかる。

よって本開発支援では、当所が保有する画像処理に関するノウハウとこれまでのシステム構築の経験を活用し、Ubuntu 搭載のミニ PC に web カメラを接続したシンプルな構成で、図１のように金型を上部から撮影したカメラ映像から離型不良を自動で判定する支援ツールを試作した。

２．実装する画像処理アルゴリズム

２．１．離型不良検出条件の設定

本支援ツールは、カメラ映像内から任意の判定対象領域を複数設定し、カメラ映像の色情報を、コンピューター上で使われる RGB 形式(赤・緑・青)から人の感覚に近い HSV 形式(色相・彩度・明度)に変換[1)]して扱う。

検出条件の設定では、検出対象となる製品の色のレンジを色相・彩度・明度それぞれ設定した後、製品が判定対象領域にあるかを判定するための閾値をパーセントで設定する。また、金型が開いて製品が露出してから正常に離型するまでに数秒ほどかかるため、離型にかかる秒数も設定するようにした。

２．２．良否判定

カメラ画像を HSV 形式に変換し、各判定領域に良否判定を行う。それぞれの判定領域に対して、色がレンジ内に収まる画素にマスクをかけ、判定レンジ内に収まる画素数をカウントする。閾値をパーセントで設定し、判定レンジ内に収まる画素の割合が閾値を超えた場合はその領域に製品が映っていると判定する。ここで離型するのにかかる秒数を超えて製品が映り続けていた場合にその領域は NG、そうでなければその領域は OK とする。

最終的にすべての領域で NG 判定が出なければ OK 判

事業名「ものづくり中小企業開発・開拓推進事業」（応募企業　笠原工業株式会社）

定とする。

３．アプリケーション

３．１．操作画面

アプリケーションの実装では、プログラム言語は Python を使用し、GUI アプリケーション作成ライブラリ tkinter[2]、画像処理ライブラリ OpenCV[3]を用いて直感的に操作できるアプリケーションを Ubuntu 上に作成した。操作画面は図２のとおり。現場の金型をカメラで撮影し、表示モニタを金型脇の作業スペースの脇に配置する。

図２画面上側の Detection View 部分にカメラ映像及び各判定領域、判定結果を表示し、下側の Control 部分で判定機能のオンオフとプログラムの終了ができる。また、左上の設定から検出条件の設定ができる。

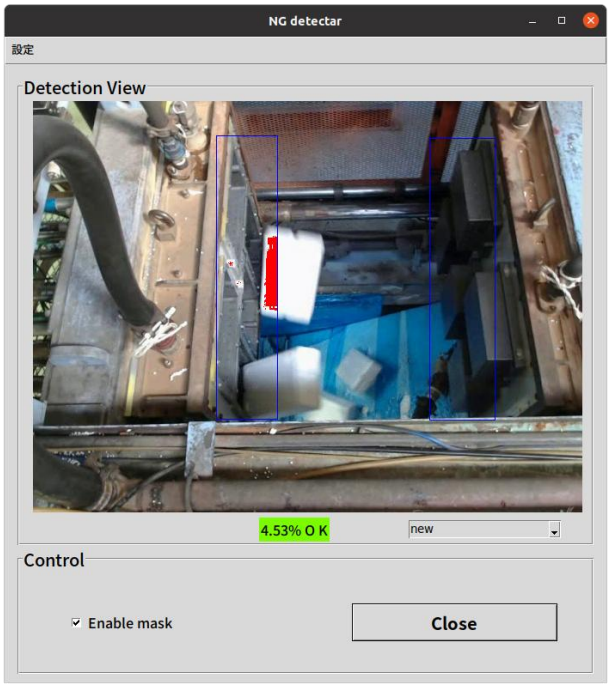

図２　操作画面

３．２．判定対象領域及び検出条件の設定

図２下側 Control 部分の Enable mask チェックを外すことで判定対象領域の設定が可能になる。Detection View 右下のプルダウンメニューで new を選択した状態でカメラ映像内の判定したい領域を矩形で囲むと、新しい判定領域が設定される。また、すでに判定領域がある場合、プルダウンメニューに番号が追加される。番号を選択した状態で同様の操作を行うと対応した判定領域を変更でき、Delete キーを押すことで対応した判定領域を削除できる。

また、左上の設定メニューから検出設定を選択することで、検出する条件の設定ができる。条件設定画面は図３のとおり。

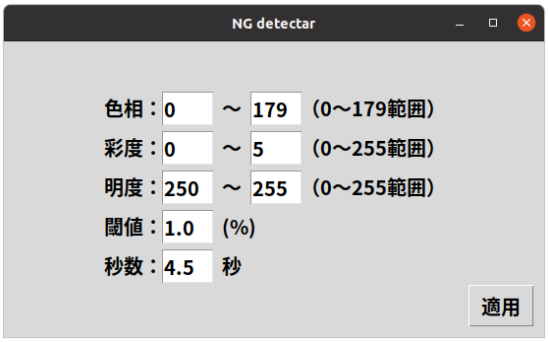

図３　条件設定画面

ここで、検出対象の色相・彩度・明度のレンジを数値で設定する。表示されているカメラ映像上で右クリックするとその場所の色相・彩度・明度が表示されるため、それを参考に設定する。また各判定領域で指定した色が何パーセントあれば対象製品があると判定するかの閾値を設定し、さらに対象製品が何秒以上判定領域に残っていたら離型不良が発生したと判定するかの秒数を指定する。

３．３．離型不良判定

図２Control 部分の Enable mask が有効化されているときは、カメラ映像からリアルタイムで判定する。

図２Detection View 内のカメラ表示部分で、矩形で表示している判定領域内のうち設定した色のレンジに収まっている部分をマスキング[4]し、赤く表示[5]している。

赤く表示された画素数を判定領域ごとにカウント[6]し、設定した閾値以上の場合はその領域内に製品が存在し、閾値未満の場合は製品が存在しないと判定する。ここで、領域内に製品が存在する状態のまま設定した秒数が経過した場合、その領域で離型不良が発生していると判定する。

この処理を各判定領域で行い、どこか一か所でも離型不良が発生していると判定した場合、図４のように赤く NG 表示を行う。また同時に NG 発生時の時刻をコンソール画面上に表示し、デスクトップ上のフォルダに NG 判定時のカメラ画像を保存する。これにより、一目ですぐに判定結果が把握できる。

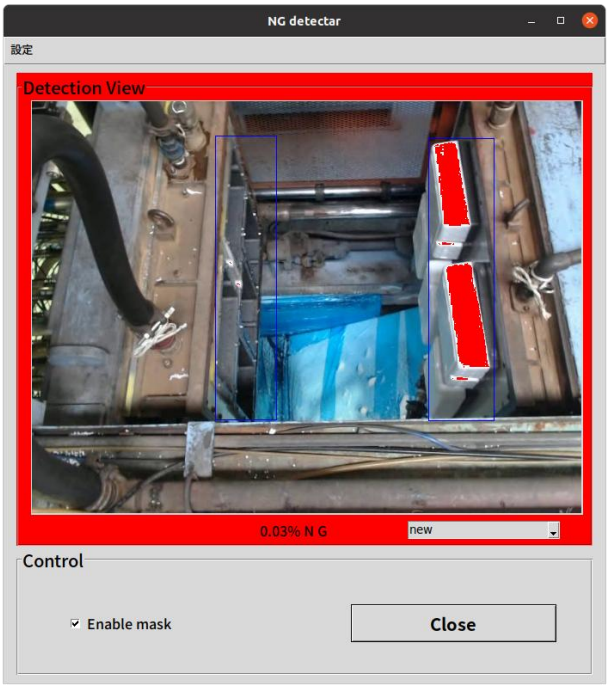

図４　NG 判定画面

４．結言

本開発支援では、離型不良の発見作業の省力化を支援するため､離型不良の自動検出システムを試作した。このシステムはPython と画像処理ライブラリ OpenCV、GUI アプリケーション作成用ライブラリ Tkinter で作成し、金型部分にカメラをセットするだけで離型不良が楽に確認できるようにした。リアルタイムで判定結果が表示され、判定対象の配置や色が変わってもシステム内で設定変更できるようにした。

事前に撮影した離型不良発生時の動画を使って検証したところ、１回誤検出したものの動画に記録された全４回の離型不良をすべて検出できた。そのため応募企業の工場内で試験導入を行い、現在検証していただいているところである。

なお、作業スペースの脇に置いた表示モニタから金型の様子を確認することができるため、とても助かるとの評価をいただいた。

今後は環境変化による誤検出対策や他製品への対応もできるようにしていきたい。

参考文献

１）OpenCV Documentation. “Changing Colorspaces”OpenCV公式ドキュメント. https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html, (参照 2025-02-24)

２）Python Software Foundation. “tkinter --- Tcl/Tk の Python インターフェース”.Python Docs. https://docs.python.org/ja/3.13/library/tkinter.html, (参照 2025-02-24)

３）OpenCV team. “OpenCV - Open Computer Vision Library”. OpenCV 公式ホームページ. https://opencv.org/, (参照 2025-02-24)

４）mryssng. ”【Python・OpenCV】色相の範囲による閾値処理・色抽出(cv2.inRange)”.codevace. https://www.codevace.com/py-opencv-inrange/, (参照 2025-02-24)

５）OpenCV Documentation. “Arithmetic Operations on Images”OpenCV公式ドキュメント. https://docs.opencv.org/4.0.0/d0/d86/tutorial_py_image_arithmetics.html, (参照 2025-02-24)

６）NumPy documentation “numpy.count_nonzero”. NumPy公式ドキュメント. https://numpy.org/doc/2.2/reference/generated/numpy.count_nonzero.html, (参照 2025-02-24)