検査工程における配線コード色判別システムの開発

Development of wire color check system for inspection process

電子・機械技術部　電子・情報科　山田昌幸　鈴木健司

現在人が行っている検品作業を省力化するため、Python 及び OpenCV を使用して、カメラから取得した映像から配線基板のコードの色を判別するシステムを開発した。システムは検査対象を保持する治具と２列のコードを両側から撮影するカメラ及び照明、さらに画像処理を行うプログラムで構成される。このシステムでは、対象をカメラの前に置くとそれぞれの位置のコードが正しいか一度に判定できる。これを現場へ試験導入し、現場での検品作業の省力化の検証作業を支援した。

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

１．緒言

申請企業の東京通信機材株式会社白河工場では、回転体に対して電力や信号を伝達することができる回転コネクタであるスリップリングを製造している。スリップリング内で使われる配線基板上には多数の配線コードが２列で実装されており（図１）、色ごとに正しい位置に配線しなければならない。

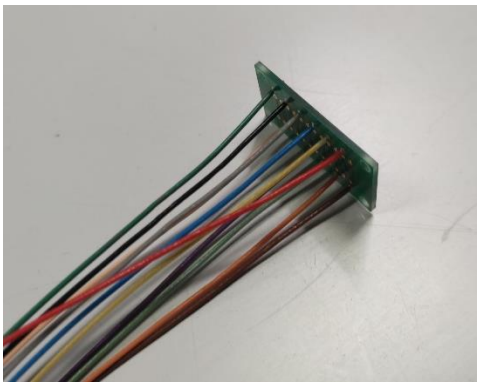

図１　配線コード基板

そのため目視による検査がされていたが、配線コード基板はサイズが小さく検査数量も多いため時間がかかり見逃しが発生していた。画像処理技術により色判別を行い、配線の検査を省力化するシステムを構築したいと考えていた。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。

そこで本開発支援では、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、図２のように２列の配線を上下から撮影したカメラ画像から配線コードの色を自動で検査する支援ツールを開発した。

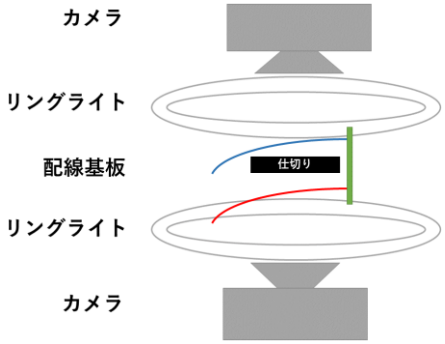

図２　概略図

２．撮像装置

２．１．全体構成

テーブルの角を挟んで支柱を２本固定し、そこにそれぞれ治具を図３のように取り付けた。

まず中央に配線コード基板を固定する治具を取り付け、それを上下から照らすようにリングライトを取り付けた。さらにリングライト中央の穴から配線コードを撮影できるようカメラを上下から取り付けた。

図３　治具全体像

２．２．配線基板固定用治具

配線コード基板を固定する治具は３Ｄプリンターで

事業名「ひとつ、ひとつ、実現するものづくり企業支援事業」（応募企業　東京通信機材株式会社白河工場）

白と黒の２通りを作成した。これを図４に示す。なお、コードに治具と同色のものがあり、その部分が治具の色に埋もれてしまうことを防ぐため白い治具で白いコードが重なる部分は黒く、黒い治具で黒いコードが重なる部分は白く変化させた。

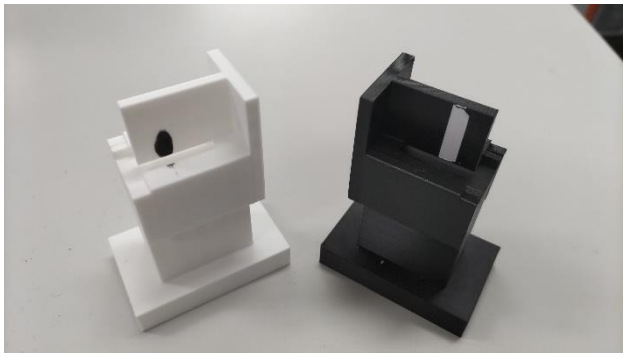

図４　配線コード基板固定用治具

３．実装する画像処理アルゴリズム

３．１．判定基準色および判定レンジの設定

作成したプログラムでは、色情報を RGB 形式(赤・緑・青)から人の感覚に近い HSV 形式(色相・彩度・明度)に変換して扱う。

両側カメラから取得した画像内にある目的の配線コード部分に対して、まず RGB 形式から HSV 形式へ変換[1]する。変換画像の H、S、V それぞれについて全画素の平均値を計算し、これを判定基準色とする。同時にそれぞれの最大値と最小値を判定レンジとする。

３．２．良否判定

まず、カメラ画像を HSV 形式に変換し、各コード毎に良否判定を行う。それぞれのコード部分に対して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。しきい値を設定し、判定レンジ内に収まる画素数がしきい値を超えた場合はそのコードは OK、超えなかった場合は NG とする。

最終的にすべてのコードで NG 判定が出なければ OK 判定とする。

４．アプリケーション

４．１．操作画面

アプリケーションの実装には、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５のとおり。

画面左側の Detection View 部分に上下両側のカメラ画像およびコード毎の判定結果をそれぞれ表示し、左下の Total Result 部分に全体の判定結果を表示している。

画面右側の Parameter Setting 部分に判定基準色を一覧で表示しており、右下の 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。

４．２．初回起動時設定

このプログラムでは、1280×960 ピクセルのカメラ画像から 640×160 ピクセルの範囲を切り出して上下にそれぞれ表示している。ここのカメラ部分の設定は配線コードの列数、OS 上で認識されている上下のカメラの番号、それぞれのカメラ画像からの切り出し位置があり、ここは設定用テキストファイルの項目を書き換えることで容易に変更できるようにした。

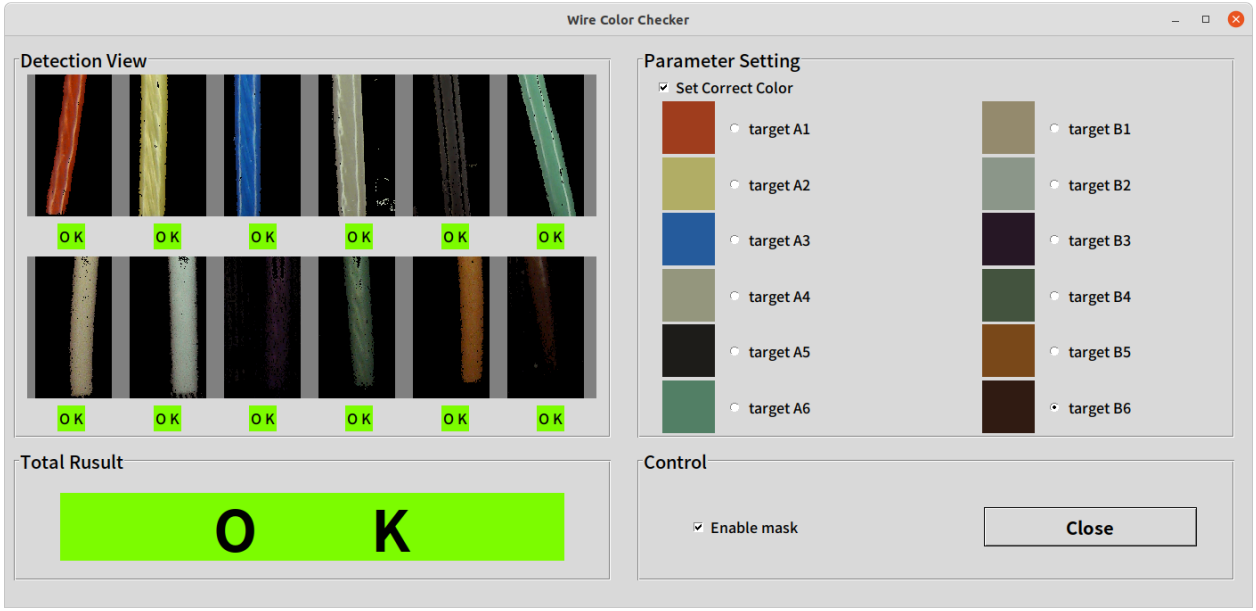

図５　操作画面

４．３．判定基準色及び判定レンジの設定

右下 Control 部分の Enable mask チェックを外し、右上 Parameter Setting の Set Correct Color のチェックを入れることで設定が可能になる。

視認性のため、各コードの基準色サンプルを右側 Parameter Setting で一覧表示しており、そこから設定するコードを選択する。targetA が上側、targetB が下側に対応しており、数字が画像左側のコードから順に割り振った。

また、配線コードを選択し配線コード画像内を枠で囲むと、判定基準色及び判定レンジが再計算・更新されるようにした。これによりマウス操作のみで色の設定が完了する。

基準色が更新されると一覧に即時反映され、Enable mask チェックを入れると更新された基準で判定するようにした。

また、基準色及び判定レンジの設定はテキスト形式で保存され、起動時に自動で読み込むようにした。

４．４．コード色判定

右下 Enable mask が有効化されているときは、リアルタイムでカメラ画像から判定する。

左上 Detection View 部分に上下カメラから切り出した画像をコード毎に分割して表示している。ここで、それぞれの判定レンジに収まっている部分をマスキング [4] し、目的のコード色に合致している画素のみを表示 [5] している。また、レンジに収まっている画素数をコード毎にカウント [6] し、設定したしきい値以下の場合は NG、超えていた場合は OK 判定とし、それぞれのコードの下部に表示する。今回はしきい値を 2000 ピクセルとした。これをすべてのコードに行い、すべて OK の場合は OK、そうでなければ NG を左下の Total Result 部分に表示するようにした。これにより、一目ですぐに判定結果が把握できる。

５．結言

本開発支援では、スリップリングの配線コードの検品作業の省力化を支援するため、コードの色の自動判別システムを作成した。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、配線コードをはんだ付けした基板をはめるだけで検品作業ができるようにした。リアルタイムで判定結果が表示され、また別の色の配線パターンを検品する際もマウス操作だけで色の設定を行えるようにした。また、コードの本数やカメラの位置などが変わった場合も設定用テキストファイルの項目を書き換えることで変更できるようにした。また、申請企業の工場内で図６のとおり設置の実演と動作確認を行い、現場に試験導入を行った。その後現場で評価をしていただいたところ、色の設定をした後の誤検出は現状ゼロであり、また製品の知識が無くても検査ができる点が有効であるとの評価を得た。

本システムによって、現場での検品作業省力化の検証に寄与するものと期待される。また、今後は検品のタクトタイムの短縮や異なる光学的環境への対応もできるようフォローしていきたい。

図６　企業での動作確認

参考文献

１）“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 2024-02-19).

２）“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 2024-02-19).

３）“OpenCV - Open Computer Vision Library”. OpenCV 公式ホームページ. https://opencv.org/, (参照 2024-02-19).

４）”【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 2024-02-19).

５）“Arithmetic Operations on Images”.OpenCV公式ドキュメント. https://docs.opencv.org/4.x/d0/d86/tutorial_py_image_arithmetics.html, (参照 2024-02-19).

６）“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 2024-02-19).