B-3871

PyFIEプログラマーズガイド

プログラム作成

☆第1版☆

はじめに

本ドキュメントは画像処理ライブラリ PyFIE を用いて、これから画像処理ソフトウェアの開発を行う方向けのガイドブックです。
Windows 64bit(x64)環境にて、Visual Studio Code を開発環境として PyFIE のインストールから画像処理の実行までの基本的な方法について実際の作業手順を例にして、PyFIE のコーディングについて解説します。

なお、本ドキュメントにおけるコーディング部のインデントには、スペース、タブは使用されていませんので、本文をコピーして使用する場合は、インデントを入力してお使いください。

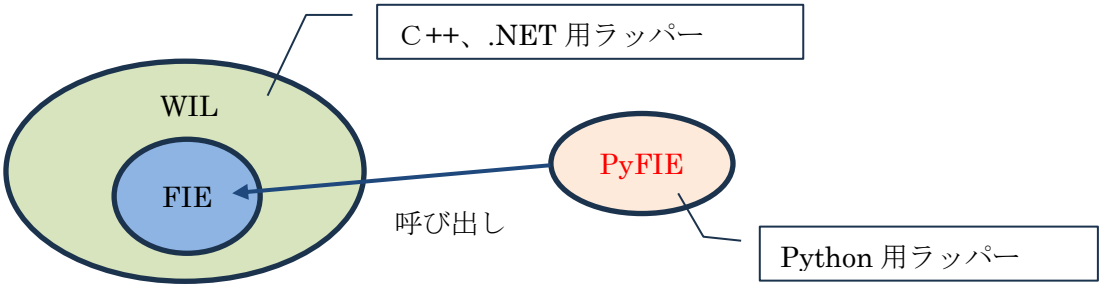
目次

1. PyFIE について

1.1 概要

「PyFIE」は、弊社画像処理アルゴリズムの根幹となるＣ言語関数群「FIE」(FAST Image Engine)をPython 上で使用可能にするラッパーライブラリです。

FIE 本体は、Windows 用ライブラリ「WIL」に同梱されている為、まずこれのインストールが必要となります。



その為、PyFIE の実行環境には、Python3.6.0 以上、および、PyFIE に対応する WIL3.1 以上の WIL ライブラリのインストールが別途必要となります。

1.2 ソフトウェア及びドキュメント

本資料内で使用するソフト及びドキュメントは、下記リンクよりダウンロード可能です。

・WIL 標準ライブラリ
https://www.fast-corp.co.jp/software_dl/jp/supportj_flv_basicpac.php?sid=164&stid=126&sdid=422

・WIL PyFIE ライブラリ
https://www.fast-corp.co.jp/software_dl/jp/supportj_flv_basicpac.php?sid=164&stid=144&sdid=424

・PyFIE リファレンス
https://www.fast-corp.co.jp/software_dl/desk/install/wil/pyfie/pyfie_doc/index.html

・FIE リファレンス
https://www.fast-corp.co.jp/software_dl/desk/install/wil/FIE-390/index.html
FIE リファレンスは、WIL をインストールすると下記にファイルが格納されます。
C:¥FAST¥WIL¥3.1.0¥Document¥FIE.chm

・WIL 導入の手引き
https://www.fast-corp.co.jp/software_dl/upload/doc/0610/pdf_j/wil_prop.pdf

上記しているように、PyFIE は FIE 関数を内部で呼び出しておりますので、各画像処理関数の詳細仕様については、FIE リファレンスマニュアルをご参照いただく必要があります。

本ドキュメント内でも関連ページについては FIE リファレンス と注釈を入れております。

2. 環境設定

2.1 必要環境

本ドキュメントの内容を実施する場合、以下の環境が必要になります。
なお、Python のインストールは完了していることを前提とします。
・Python ver3.6.0 以上
・WIL ver3.1.0.0 以上
・PyFIE
・matplotlib
・エディタ（VScode 等）

2.2 WIL のインストール

弊社 web の下記 URL(リンクは 1.2 項)より、WIL のインストーラー「WIL.msi」がダウンロードできますので、実行して WIL のインストールを行ってください。

・WIL 標準ライブラリ
https://www.fast-corp.co.jp/software_dl/jp/supportj_flv_basicpac.php?sid=164&stid=126&sdid=422

詳細については、弊社 web の下記 URL(リンクは 1.2 項)にある「WIL 導入の手引き」の 2.3.1 項を参照してください。

・WIL 導入の手引き
https://www.fast-corp.co.jp/software_dl/upload/doc/0610/pdf_j/wil_prop.pdf

2.3 PyFIE のインストール

Python3 系では 3.4 以降、Python と同時に pip パッケージマネージャもインストールされておりますので、ここでは PyFIE を pip パッケージマネージャにてインストールする方法を紹介します。
なお、python に関して、ver3.6.0 より前のバージョンがインストールされている場合、PyFIE のインストールに失敗する事がありますので、旧バージョンの python に関しましては予め削除してください。

下記のコマンドにて、pip パッケージマネージャによる PyFIE のインストールが行われます。

```
pip install -U fast-pyfie
```

※PyFIE 3.8.0.5 より、パッケージ名を"pyfie"から"fast-pyfie"に変更しました。
この変更のため、バージョン 3.7.2.4 以前の PyFIE から新しい PyFIE へアップデートする際には、一度古い PyFIE をアンインストールする必要があります。

下記のコマンドにて、正常にインストールされているか確認してみましょう。

```
pip list
```

正常にインストールされた場合、次ページの様に表示されます。
なお、「X.X.X.X」には、インストールした PyFIE のバージョンが表示されます。

```
Package            Version
------------------ --------
 ・・・             ・・・
fast-pyfie         X.X.X
```

2.4 Matplotlib のインストール

Matplotlib は、Python 用のグラフ描画ライブラリになります。
Matplotlib が使用可能な環境では、Pyplot を利用した PyFIE のプロット機能が有効となり、画像オブジェクトや構造体によるプロットを行うことができます。
なお、対応バージョンは、Matplotlib 2.0.0 以上になります。

下記のコマンドにて、pip パッケージマネージャによる Matplotlib のインストールが行われます。
なお、本ドキュメントでは、結果の表示に Matplotlib を使用していますので、インストールを推奨します。

```
pip install matplotlib
```

インストールを行った場合は、下記のコマンドにて、正常にインストールされているか確認してみましょう。

```
pip list
```

正常にインストールされた場合、下記の様に表示されます。
なお、「X.X.X」には、インストールした Matplotlib のバージョンが表示されます。

```
Package            Version
------------------ --------
 ・・・             ・・・
matplotlib         X.X.X
```

Matplotlib によるプロット機能の詳細仕様については、弊社 web の下記 URL(リンクは 1.2 項)にある「WIL PyFIE ライブラリ説明書」の「チュートリアル→Matplotlib によるプロット機能」をご参照ください。

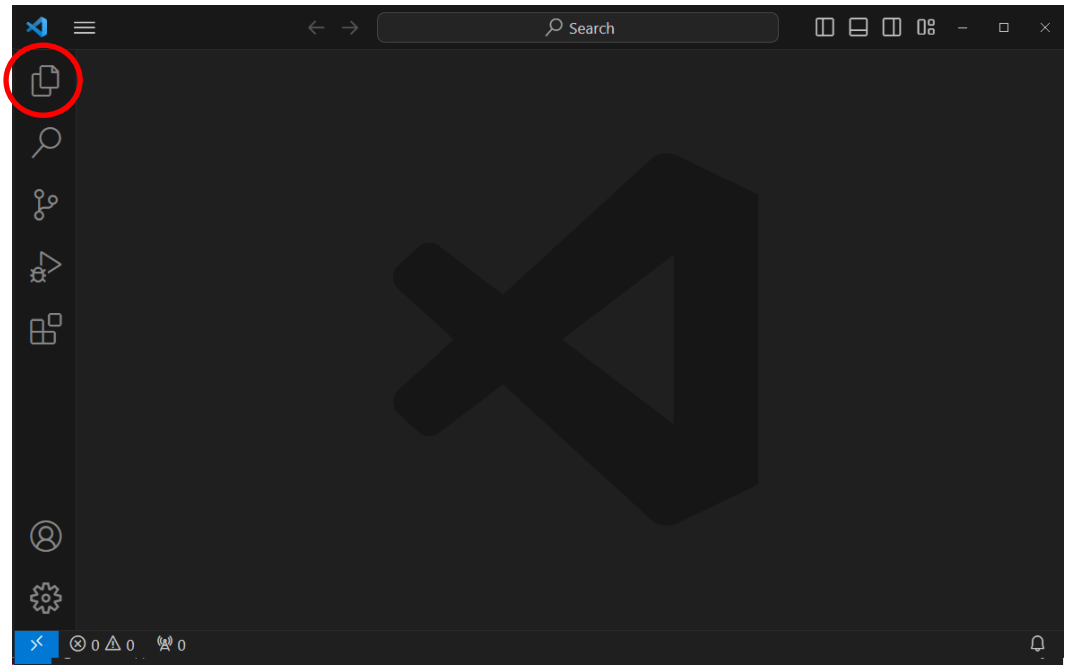
```
・WIL PyFIE ライブラリ説明書
https://www.fast-corp.co.jp/software_dl/desk/install/wil/pyfie/pyfie_doc/index.html
```

3. プロジェクト作成、初期設定

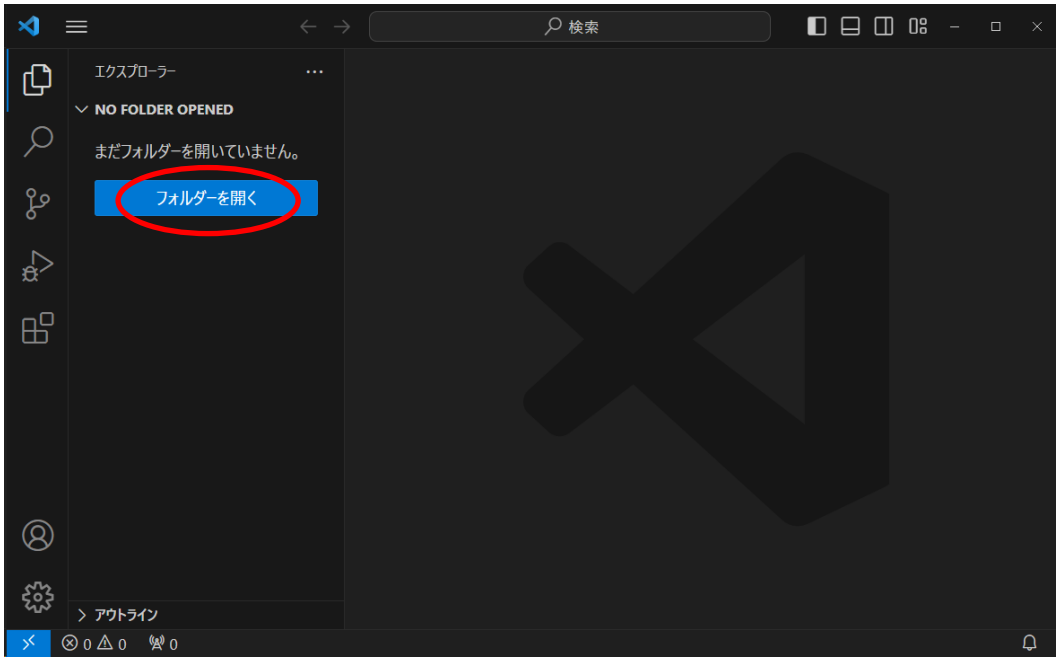
本ドキュメントでは、Visual Studio Code（以下、VSCode 表記します）を使用した python を実行するまでの手順について説明致します。

3.1 作業フォルダ、py ファイルの作成

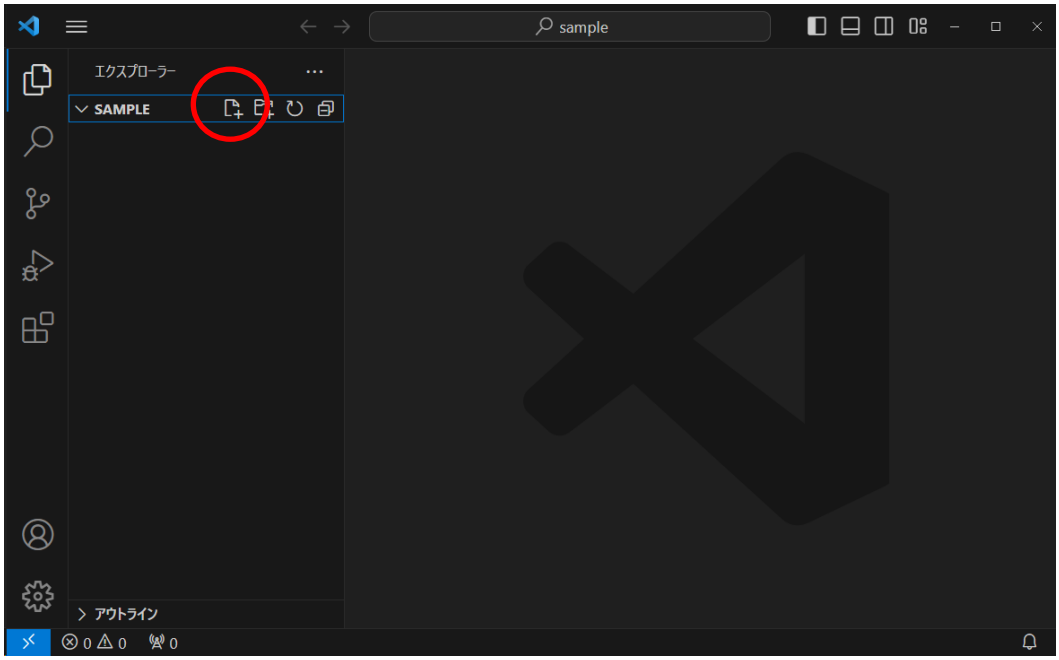
①VSCode の下記赤丸部のウィンドウ左端のアクティビティバーの一番上にある、エクスプローラーを選択してください。



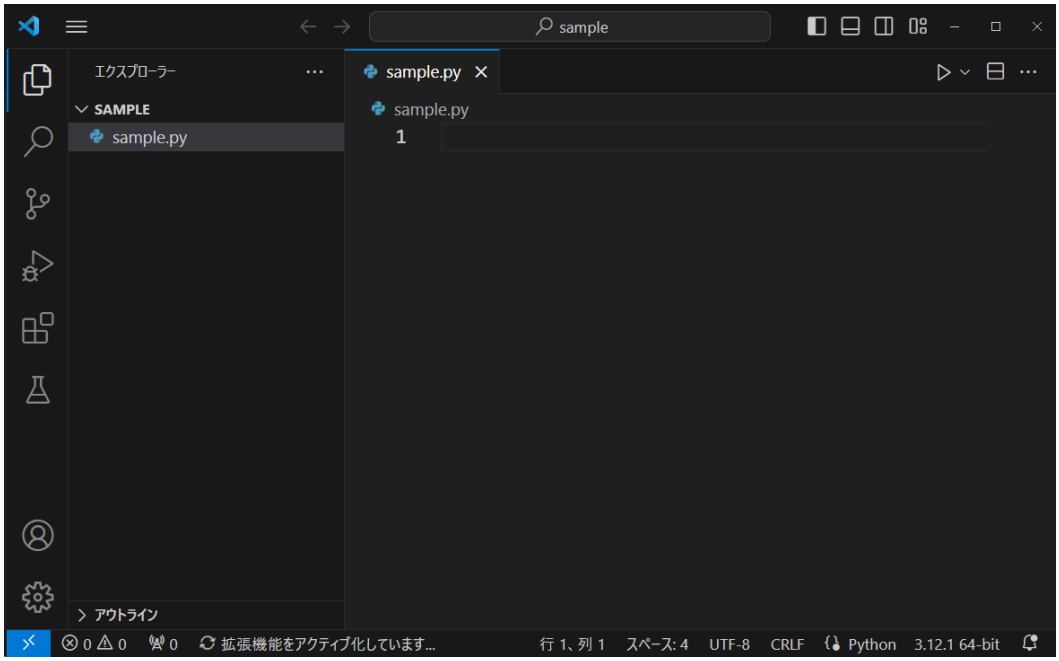
②この画面の下記赤丸部「フォルダを開く」を選択し、「C:¥sample」等の任意の場所に作業フォルダを作成してください。



③この画面の下記赤丸部「新しいファイル…」を選択し、「sample.py」等の py ファイルを作成してください。



④本ドキュメントでは、以降「sample.py」に記述していきます。



3.2 PyFIE のインポート

Python で PyFIE を使用するには、インポートを行う必要があります。
PyFIE のモジュール名は pyfie となっておりますので、Python から下記例のようにインポートすることができます。
なお、本ドキュメントでは、以降のコーディングに関して、下記のように PyFIE のインポートが行われていることを前提とします。

```
import pyfie
```

上記の方法でインポートを行った場合、PyFIE は下記のようにモジュール名 pyfie を使って使用することになります。

```
import pyfie
# モジュール名 pyfie を使用して fnFIE_img_root_alloc()関数を呼び出し
img = pyfie.fnFIE_img_root_alloc( ... )
```

※PyFIE を使用する際にはライセンスキーが、PC に接続され、認識されている必要があります。

3.3 モジュール名の短縮方法

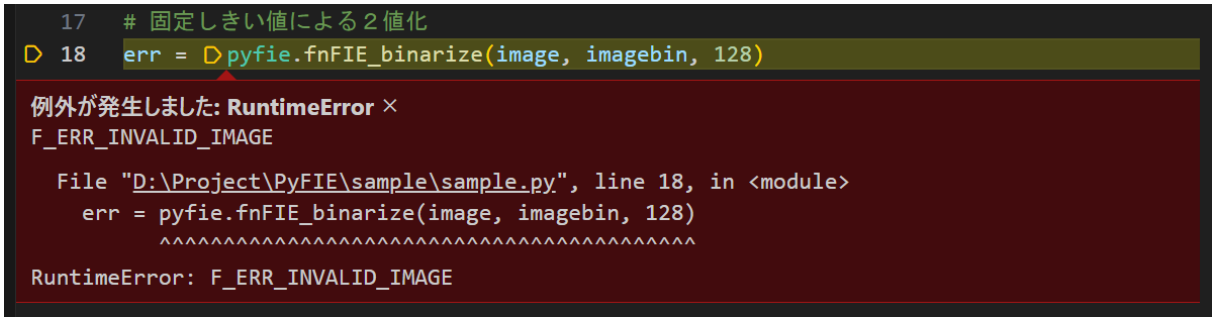
"pyfie" という 5 文字を短縮したい場合は下記のようにインポートを行うこともできます。

```
# PyFIE に任意の名前（ここでは fie）をつけてインポート
import pyfie as fie
img = fie.fnFIE_img_root_alloc( ... )
```

3.4 例外を発生させる機能の有効化

デバッグ実行時に、PyFIE 関数でエラーが有った際に、その関数の場所で例外を発生させる機能があります。この機能を使用する場合、インポートの次に以下の記述を行って下さい。

```
# 利便のため PyFIE 関数が返すエラーコードに応じて例外を発生させる機能を有効化
pyfie.ctrl.enable_f_err_exception(True)
```



3.5 FIE ライブラリの初期化について

FIE ライブラリの使用時には、fnFIE_setup()関数によるライブラリ全体の初期化処理が行われている必要がありますが、PyFIE はインポートの際に内部で fnFIE_setup()の呼び出しが行われますので、明示的に fnFIE_setup()の呼び出しを行う必要はありません。

4. 画像読み込み・表示

本項では、画像ファイルを読み込み、画面表示する手順を解説します。

※本項では、弊社 Web からダウンロードできる WIL PyFIE ライブラリの
「¥pyfie_doc¥_images」に同梱されている「captured.png」を処理対象画像として説明を行います。
※WIL PyFIE ライブラリは、弊社 web の下記 URL(リンクは 1.2 項)よりダウンロードできます。
なお、記載の URL よりダウンロードいただくには会員登録が必要になりますので、会員登録を
行っていただきますようお願い致します。

・WIL PyFIE ライブラリ
https://www.fast-corp.co.jp/software_dl/jp/supportj_flv_basicpac.php?sid=164&stid=144&sdid=424

4.1 画像の読み込みと表示

下記の赤字部分を参考に、画像読み込みと、確認の為の画像を表示する処理を追加してください。

```
# 画像読み込み
image = pyfie.imread("captured.png")  # …(1)
# 画像オブジェクトのプロット
image.imshow()  # …(2)
# プロットした画像の表示
pyfie.plot_show()  # …(3)
```

(1) 表示対象画像を読み込みます。
なお、画像ファイルのパスはお使いの環境に合わせて記述してください。
(2) 読み込んだ画像をプロットします。
(3) プロットした画像を表示します。

上記を実行しますと、 [画像 1]が表示されます。

[画像 1]

5. 2 値ブローブ解析

本項では、画像処理手法の一つである２値ブローブ解析の手順について、以下の順番にて解説します。

・画像の準備
・2 値化処理
・2 値ブローブ解析
・ブローブ選別フィルタ処理
・2 値ブローブ解析結果のターミナル表示
・2 値ブローブ解析結果画像の表示
・メモリの開放
・プログラムの実行

なお、画像の準備については、「4 画像読み込み・表示」を参照の上、事前にコードを追加してあるものとします。

5.1 処理画像格納用メモリの確保

2 値ブローブ解析にて扱う入力画像は、2 値画像である必要がある為、画像メモリに格納した画像が濃淡画像やカラー画像の場合、2 値画像へ変換する必要があります。
本項で使用する画像「captured.png」は、濃淡画像ですので、変換後の 2 値画像を格納する画像メモリに関するコーディングを行います。

2 値化関数では、入出力に同一の画像メモリを指定する事はできませんので、読み込んだ画像サイズに合わせた 2 値画像用のメモリを新たに確保する必要があります。
下記の赤字部分を参考に、2 値画像格納用メモリを確保する処理を追加してください。

```
# ２値画像格納用メモリ
imagebin = pyfie.fnFIE_img_root_alloc(pyfie.F_IMG_BIN, 1, image.width,
                                      image.height)  # …(4)
```

(4) 変換後の２値画像用の画像メモリを確保します。

FIE リファレンス

目次→モジュール→FIE module→画像オブジェクト→関数→fnFIE_img_root_alloc

5.2 2 値化処理

濃淡画像をしきい値にて 2 値画像に変換します。

①下記の赤字部分を参考に、処理対象画像を 2 値化する処理を追加してください。

```
# 固定しきい値による２値化
err = pyfie.fnFIE_binarize(image, imagebin, 128)  # …(5)
```

(5) 画像「image」に対し、固定しきい値による 2 値化処理を実行します。
本ドキュメントでは、「captured.png」に合わせて「128」のしきい値を設定しています。

FIE リファレンス

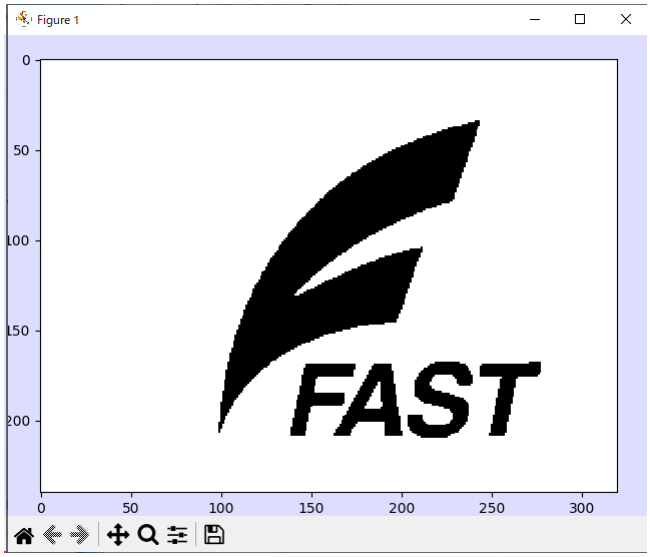
目次→モジュール→FIE module→画像フィルタ→２値化／セグメンテーション→関数→fnFIE_binarize

②2 値化処理が正常に行われたのか、表示して確認してみましょう。
また、ファイル保存する一例も記述します。
下記の赤字部分を参考に、ファイル表示、ファイル保存する処理を追加してください。

```
# 画像オブジェクトのプロット
imagebin.imshow()  # …(6)
# プロットした画像の表示
pyfie.plot_show()  # …(7)
# imagebin画像を保存
pyfie.imwrite("imagebin.png", imagebin)  # …(8)
```

(6) 読み込んだ画像をプロットします。
(7) プロットした画像を表示します。
(8) 2 値化処理した画像「imagebin」をファイル名「imagebin.png」として保存します。

上記を実行しますと、[画像 2]が表示されます。



[画像 2]

また、保存した「imagebin.png」は、[画像 3]になります。

[画像 3]

5.3 2 値ブローブ解析

2 値ブローブ解析とは、2 値画像内の白塊または黒塊の図形を解析し、その特徴量を取得する手法です。

本項では、前項で 2 値化した画像に対して、設定したブローブ解析処理パラメータによる、2 値ブローブ解析の実行に関するコーディングを行います。

①下記の赤字部分を参考に、ブローブ解析処理パラメータの設定を追加してください。

```
# ブローブ解析パラメータの設定
params = pyfie.F_MEASURE_PARAMS(
    0,   # 最大ラン数
    0,   # 最大ブローブ数
    0,   # 最大行数
    pyfie.F_MEASURE_BLACKFG,   # 対象色モード
    8,   # 連結モード
    0,   # 事前計算特徴量指定
    False  # 結果維持オプション
)  # …(9)
# 対象色モードの表示
print("color_mode =", params.color_mode)  # …(10)
```

(9) 2 値ブローブ解析にて使用する、ブローブ解析処理パラメータを設定します。

(10) ターミナルに対象色モードを表示します。

FIE リファレンス

目次→モジュール→FIE module→２値ブローブ解析→実行→データ構造→F_MEASURE_PARAMS

実行すると、ターミナルに下記の様に表示されます。

```
color_mode = 2
```

②下記の赤字部分を参考に、2 値画像とブローブ解析処理パラメータを使用して、2 値ブローブ解析する処理を追加してください。

```
# 2値ブローブ解析の実行
err = pyfie.INT()  # …(11)
result = pyfie.fnFIE_measure_execute(imagebin, (0, 0), params, err.ref)  # …(12)
# エラーコード番号の表示
print("err =", err)  # …(13)
```

(11) エラーコード用変数の初期化をします。
(12) 2 値画像「imagebin」とブローブ解析処理パラメータ「params」を使用して、2 値ブローブ解析を実行します。
(13) ターミナルにエラーコード番号を表示します。

FIE リファレンス

目次→モジュール→FIE module→２値ブローブ解析→実行→関数→fnFIE_measure_execute

実行すると、ターミナルに下記の様に表示されます。

```
err = 0
```

試しに、fnFIE_measure_execute の引数「imagebin」を「image」に変更して実行すると、ターミナルに下記の様に「F_ERR_INVALID_IMAGE（不正な画像エラー）」のエラーコード番号である「-20002」が表示されます。

```
err = -20002
```

5.4 ブローブ選別フィルタ処理

前項にて、2 値ブローブ解析を実行し、正常に終了した場合は、２値ブローブ解析結果ハンドル（F_MEASURE_RESULT オブジェクト）を返し、解析結果は「result」に格納されます。

ただし、その結果には、ノイズ等を含む画像内にある全てのブローブの解析結果が格納されていますので、その中から必要とするブローブを選別する必要があります。

面積値、縦横比などのフィルタを使用することにより、任意のブローブのみ取得できますので、本項では、得られた解析結果に対するフィルタリング処理に関するコーディングを行います。

①下記の赤字部分を参考に、フィルタ配列の設定を追加してください。

本項では、面積値が 100 以上、5000 以下、X 座標の最小値が 150 以上、300 以下のブローブを選別するフィルタ配列を設定しています。

```
# フィルタ配列の設定
filter_num = 2  # …(14)
filters = pyfie.F_MEASURE_FILTER_RANGE.ARRAY(filter_num)  # …(15)
filters.value = (
    (pyfie.F_FEATURE_AREA, 100, 5000),
    (pyfie.F_FEATURE_XMIN, 150, 300)
)  # …(16)
# 配列要素の 0 番目を表示
print(filters[0])  # …(17)
```

(14) フィルタ配列の要素数を設定します。

(15) フィルタ配列の構造体を「filter_num」分、作成します。

(16) フィルタ配列の値を、面積値が 100 以上、5000 以下、X 座標の最小値が 150 以上、300 以下に設定します。

(17) ターミナルに配列要素の 0 番目を表示します。

FIE リファレンス

目次→モジュール→FIE module→２値ブローブ解析→実行結果操作→データ構造→F_MEASURE_FILTER_RANGE

実行すると、ターミナルに下記の様に表示されます。

```
F_MEASURE_FILTER_RANGE :
    enum type = 17
    DOUBLE min = 100.0
    DOUBLE max = 5000.0
```

②下記の赤字部分を参考に、前項で作成したフィルタ配列を使用して、フィルタリングしたブローブ番号配列を取得する処理を追加してください。

```
# フィルタリングしたブローブ番号配列の取得
blob_numbers = pyfie.UINT.PTR()  # …(18)
blob_num = pyfie.UINT()  # …(19)
pyfie.fnFIE_measure_get_list(
    result, filters, filter_num, blob_numbers.ref, blob_num.ref
)  # …(20)
# フィルタリング後のブローブの要素数を表示
print("blob_num =", blob_num)  # …(21)
```

(18) フィルタリングしたブローブ番号配列取得用の配列ポインタを確保します。
(19) フィルタリングしたブローブの要素数を取得する変数を確保します。
(20) 2値ブローブ解析結果「result」とフィルタ配列「filters」を使用して、ブローブ番号配列を取得します。
(21) ターミナルにフィルタリング後のブローブの要素数を表示します。

FIE リファレンス

目次→モジュール→FIE module→２値ブローブ解析→実行結果操作→関数→fnFIE_measure_get_list

実行すると、ターミナルに下記の様に表示されます。

```
blob_num = 3
```

5.5 2 値ブローブ解析結果のターミナル表示

前項までで得られた解析結果に対してフィルタリング処理を行い、選別したブローブ番号配列が得られたので、そのブローブの重心情報をターミナル表示するコーディングを行います。

下記の赤字部分を参考に、ブローブの重心をターミナル表示する処理を追加してください。

```
# 重心座標を受け取るための変数を用意
center_x = pyfie.DOUBLE()  # …(22)
center_y = pyfie.DOUBLE()  # …(23)
# 2 値ブローブ解析結果の表示
for i in range(blob_num):  # …(24)
    # 重心の取得
    pyfie.fnFIE_measure_get_center(
    result, blob_numbers[i], center_x.ref, center_y.ref
    )  # …(25)
    print("重心 (", center_x, ",", center_y, ") ")  # …(26)
```

(22) ブローブ重心の x 座標を受け取る変数を用意します。
(23) ブローブ重心の y 座標を受け取る変数を用意します。
(24) 取得したブローブの要素数分、表示処理を繰り返します。
(25) 指定したブローブ番号の重心座標を取得します。
(26) ターミナルに重心座標を表示します。

FIE リファレンス

目次→モジュール→FIE module→２値ブローブ解析→特徴量計算→関数→fnFIE_measure_get_center

実行すると、ターミナルに下記の様に表示されます。
なお、ターミナルの結果は「S」「T」「A」の文字の重心が順番に表示されます。

```
重心 ( 222.0036855036855 , 188.66216216216216 )
重心 ( 258.3614232209738 , 182.9250936329588 )
重心 ( 184.96241610738255 , 190.71543624161072 )
```

5.6 2 値ブローブ解析結果画像の表示

2 値ブローブ解析結果をプロットした画像を表示する例を下記に記述します。
表示する場合は、下記の赤字部分を参考に、画像表示の処理を追加してください。

```
# 画像オブジェクトのプロット
imagebin.imshow()  # …(27)
# 結果をプロット
pyfie.mpltools.plot_measure_results(result, blob_numbers, blob_num)  # …(28)
# プロットした画像の表示
pyfie.plot_show()  # …(29)
```

(27) 画像をプロットします。
(28) 2 値ブローブ解析結果「result」、ブローブ番号配列「blob_numbers」、ブローブ要素数「blob_num」を使用して、プロットを行います。
(29) プロットした画像を表示します。

5.7 メモリの開放

PyFIE 関数内部で確保され返されるメモリ領域（FHANDLE を除く）は Python のガベージコレクション対象となりませんので、ユーザーが解放処理を行う必要があります。
本項では、PyFIE 関数内部で確保されたメモリ領域の開放に関するコーディングを行います。
下記の赤字部分を参考に、メモリ開放処理を追加してください。

```
# メモリの解放、ブローブ番号配列を解放
blob_numbers.fnOAL_free()  # …(23)
```

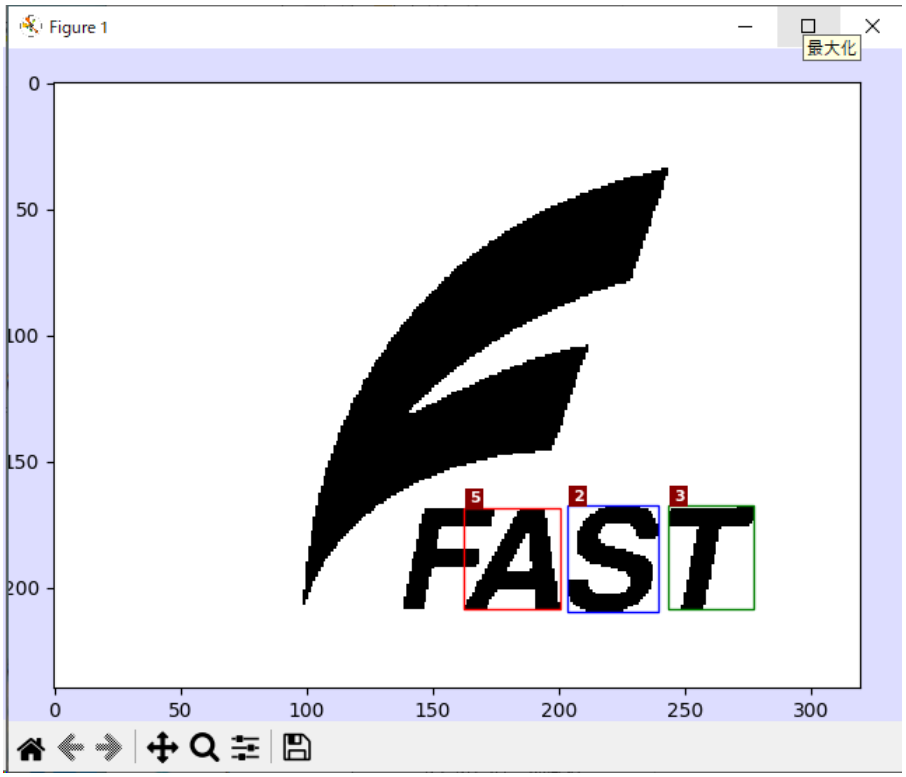
(23) fnFIE_measure_get_list()により確保されたブローブ番号配列を開放します。

FIE リファレンス

目次→モジュール→FIE module→２値ブローブ解析→実行結果操作→関数→fnFIE_measure_get_list

5.8 プログラムの実行

以上で、2 値ブローブ解析のサンプルプログラムのコーディングは終了です。
実行して、結果を確認しましょう。
実行すると、結果画像として[画像 4]が表示されます。



[画像 4]

6. グレイサーチ(正規化相関サーチ：GS2)

本項では、画像処理手法の一つであるグレイサーチ(正規化相関サーチ：GS2)の手順について、以下の順番にて解説します。

・画像の準備
・パタンオブジェクトの生成
・正規化相関サーチ(グレイサーチ)
・グレイサーチ結果のターミナル表示
・グレイサーチ結果画像の表示
・プログラムの実行

6.1 画像の準備

グレイサーチを実行するためには、パターン画像とサーチ対象画像が必要ですので、ここではそれらの画像の準備に関するコーディングを行います。

※本項では、以降、弊社 Web からダウンロードできる WIL PyFIE ライブラリの「¥pyfie_doc¥_images」に同梱されている「gs_pattern.png」をサーチパターン生成用画像、「gs_target.png をサーチ対象画像として説明を行います。

※WIL PyFIE ライブラリは、弊社 web の下記 URL(リンクは 1.2 項)よりダウンロードできます。なお、記載の URL よりダウンロードいただくには会員登録が必要になりますので、会員登録を行っていただきますようお願い致します。

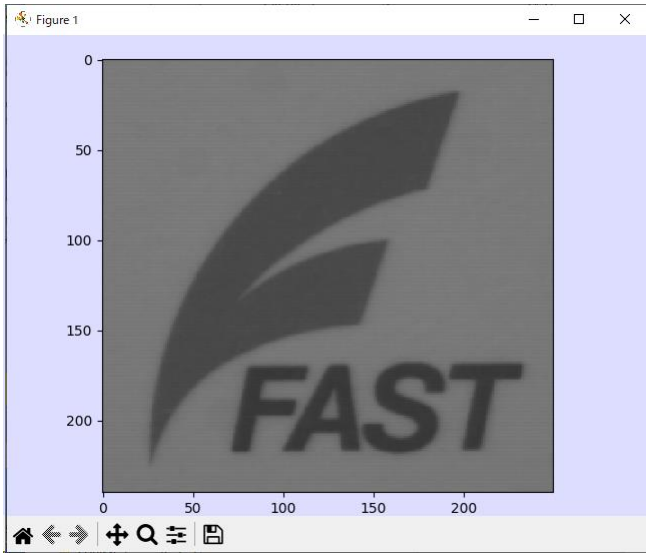
・WIL PyFIE ライブラリ
https://www.fast-corp.co.jp/software_dl/jp/supportj_flv_basicpac.php?sid=164&stid=144&sdid=424

①サーチパターン生成用画像を読み込みます。
また、確認の為に読み込んだ画像の表示処理も行います。
下記の赤字部分を参考に、画像読み込みと、確認の為の画像を表示する処理を追加してください。

```
# 画像読み込み
pattern = pyfie.imread("gs_pattern.png")  # …(1)
# 画像オブジェクトのプロット
pattern.imshow()  # …(2)
# プロットした画像の表示
pyfie.plot_show()  # …(3)
```

(1) サーチパターン生成用画像を読み込みます。
なお、画像ファイルのパスはお使いの環境に合わせて記述してください。
(2) 読み込んだ画像をプロットします。
(3) プロットした画像を表示します。

上記を実行しますと、次ページの[画像 5]が表示されます。



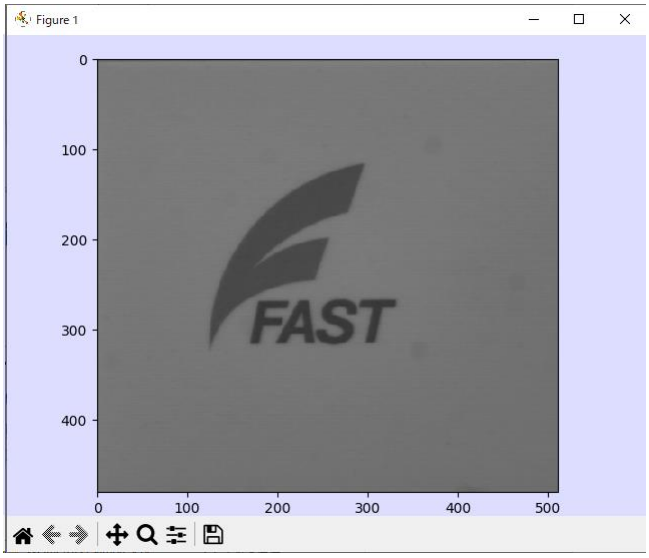
[画像 5]

②サーチ対象画像を読み込みます。
また、確認の為に読み込んだ画像の表示処理も行います。
下記の赤字部分を参考に、画像読み込みと、確認の為の画像を表示する処理を追加してください。

```
# 画像読み込み
target= pyfie.imread("gs_target.png")  # …(4)
# 画像オブジェクトのプロット
target.imshow()  # …(5)
# プロットした画像の表示
pyfie.plot_show()  # …(6)
```

(4) サーチ対象画像を読み込みます。
なお、画像ファイルのパスはお使いの環境に合わせて記述してください。
(5) 読み込んだ画像をプロットします。
(6) プロットした画像を表示します。

上記を実行しますと、[画像 6]が表示されます。



[画像 6]

6.2 パタンオブジェクトの生成

グレイサーチを実行するために必要なグレイサーチパターンオブジェクトの生成に関するコーディングを行います。

下記の赤字部分を参考に、グレイサーチパターンオブジェクトを生成する処理を追加してください。

```
# グレイサーチパターンオブジェクトの生成
gs_pattern = pyfie.fnFIE_gs2_pattern_alloc(
    pattern, None, pattern.width / 2 * 100, pattern.height / 2 * 100,
    pyfie.F_COMP_MODE_SMOOTH, None
)  # …(7)
# グレイサーチパターンオブジェクトのハンドルを表示
print(gs_pattern)  # …(8)
```

(7) 画像「pattern」からグレイサーチパターンオブジェクトを生成します。
(8) ターミナルにグレイサーチパターンオブジェクトのハンドルを表示します。

FIE リファレンス

目次→モジュール→FIE module→サーチ→グレイサーチ(正規化相関サーチ)→パタン操作→関数→fnFIE_gs2_pattern_alloc

実行すると、ターミナルに下記の様に表示されます。

```
FHANDLE: F_OBJID_GS_PATTERN
```

6.3 グレイサーチ(正規化相関サーチ：GS2)

前項で生成したグレイサーチパターンによる、グレイサーチの実行に関するコーディングを行います。

①下記の赤字部分を参考に、グレイサーチオブジェクトの生成を追加してください。

```
# グレイサーチオブジェクトの生成
hgs = pyfie.fnFIE_gs2_alloc(0, 0, 0)  # …(9)
# グレイサーチオブジェクトのハンドルを表示
print(hgs)  # …(10)
```

(9) グレイサーチの作業領域となるグレイサーチオブジェクトを生成します。
(10) ターミナルにグレイサーチオブジェクトのハンドルを表示します。

FIE リファレンス

目次→モジュール→FIE module→サーチ→グレイサーチ(正規化相関サーチ)→実行→関数→fnFIE_gs2_alloc

実行すると、ターミナルに下記の様に表示されます。

```
FHANDLE: F_OBJID_GS
```

②下記の赤字部分を参考に、サーチウインドウの生成を追加してください。

```
# サーチウインドウの生成
search_window = pyfie.BOX_T((0, 0), (target.width - 1, target.height - 1))  # …(11)
# サーチウインドウの終了点座標を表示
print(search_window.ed)  # …(12)
```

(11) 画像オブジェクトの中でサーチを行う処理範囲に使用する矩形を生成します。
サーチ対象画像の中に納まるように設定してください。
(10) ターミナルにサーチウインドウの終了点座標を表示します。

実行すると、ターミナルに下記の様に表示されます。

```
PNT_T :
    INT x = 511
    INT y = 479
```

③下記の赤字部分を参考に、サーチ結果の保存先の生成を追加してください。

```
# サーチ結果の保存先を生成
max_result_num = 5  # …(13)
result = pyfie.F_GS_RESULT.ARRAY(max_result_num)  # …(14)
result_num = pyfie.INT()  # …(15)
# 配列要素の 0 番目を表示
print(result[0])  # …(16)
```

(13) サーチ結果を取得する個数用の変数を作成します。
本ドキュメントでは 5 個に設定します。
(14) サーチ結果を格納する配列を「max_result_num」分、生成します。
(15) 見つかったサーチ結果の個数を受け取る変数を用意します。
(16) ターミナルに配列要素の 0 番目を表示します。

FIE リファレンス

目次→モジュール→FIE module→サーチ→グレイサーチ(正規化相関サーチ)→実行→データ構造→F_GS_RESULT

実行すると、ターミナルに下記の様に表示されます。

```
F_GS_RESULT :
    INT x = 0
    INT y = 0
    INT score = 0
```

④下記の赤字部分を参考に、前項で作成したグレイサーチパターンを使用して、グレイサーチを実行する処理を追加してください。

```
# グレイサーチの実行
pyfie.fnFIE_gs2_search_enforce2(
    hgs,  # グレイサーチオブジェクト
    gs_pattern,  # グレイサーチパターンオブジェクト
    target,  # サーチ対象となる画像オブジェクト
    search_window,  # サーチウインドウ
    5000,  # 途中相関値
    6000,  # 最終相関値
    False,  # サーチウインドウ周囲接触フラグ
    False,  # 反転パターン検出モードフラグ
    0,  # 同一解範囲幅
    0,  # 同一解範囲高さ
    2,  # サーチ開始圧縮度
    0,  # サーチ終了圧縮度
    pyfie.F_GS2_SUBPXL_NEIB_8,  # 近傍
    result,  # サーチ結果
    max_result_num,  # サーチ結果取得個数
    result_num  # サーチ結果の個数
)  # …(17)
# サーチ結果数の表示
print("result number =", result_num)  # …(18)
```

(17) グレイサーチパターン「gs_pattern」、サーチ対象画像「target」を使用し、上記パラメータにてグレイサーチを実行します。
(18) サーチ結果の個数を表示します。

FIE リファレンス

目次→モジュール→FIE module→サーチ→グレイサーチ(正規化相関サーチ) →実行関数→fnFIE_gs2_search_enforce2
また、パラメータの詳細仕様については、同カテゴリ内の「パラメータ解説」をご覧ください。

実行すると、ターミナルに下記の様に表示されます。

```
result number = 1
```

6.4 グレイサーチ結果のターミナル表示

前項までで得られたサーチ結果の検出個数、座標情報、スコアをターミナル表示するコーディングを行います。

下記の赤字部分を参考に、サーチ結果情報をターミナル表示する処理を追加してください。

```
# サーチ結果の表示
for i in range(result_num):  # …(19)
    print("#{}: x = {}, y = {}, score = {}".format(i, result[i].x / 100,
          result[i].y / 100, result[i].score))  # …(20)
```

(19) サーチ結果の個数分、表示処理を繰り返します。
(20) 結果格納配列の番号、座標情報、スコアを表示します。

実行すると、ターミナルに下記の様に表示されます。

```
#0: x = 223.0, y = 218.0, score = 9999
```

6.5 グレイサーチ結果画像の表示

グレイサーチ結果をプロットした画像を表示する例を下記に記述します。
表示する場合は、下記の赤字部分を参考に、画像表示の処理を追加してください。

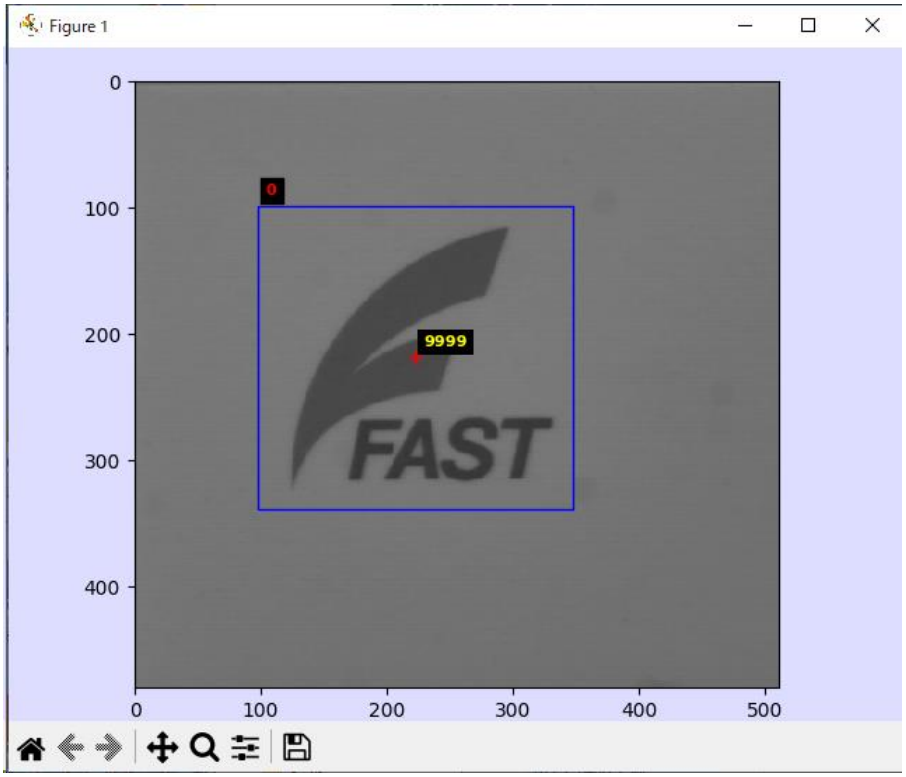
```
# 画像オブジェクトのプロット
target.imshow()  # …(21)
# 結果をプロット
pyfie.mpltools.plot_gs2_results(gs_pattern, result, result_num)  # …(22)
# プロットした画像の表示
pyfie.plot_show()  # …(23)
```

(21) 表示対象画像をプロットします。
(22) グレイサーチパターンオブジェクト「gs_pattern」、サーチ結果「blob_ result」、
サーチ結果個数「blob_num」を使用して、プロットを行います。
(23) プログラムを止めて、結果をプロットした画像を表示します。

6.6 プログラムの実行

以上で、グレイサーチのサンプルプログラムのコーディングは終了です。
実行して、結果を確認しましょう。

実行すると、結果画像として[画像 7]が表示されます。



[画像 7]