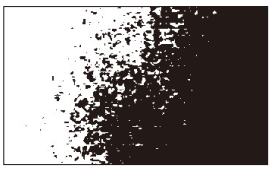
図1　2値化した画像の例

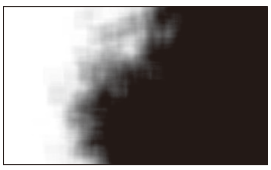
図2　ぼかし変換した画像の例

画像をぼかすということは，きれいな映像や画像をわざわざ劣化させることになります．写真や映像の分野では，明確な目的がなければこのような作業を行うことはないと思いますが，画像解析の分野では頻繁に行われます．**図1**は，とある画像の一部を拡大したものです，2値化した画像は細かいノイズが見えますが，ぼかし変換により**図2**のようにノイズの減った滑かな画像になります．

5-1 周辺画素の平均を取るブラー

プログラム名：Blur.py
CPU版 https://interface.cqpub.co.jp/5-1Blur-py/

CPU版

● 基本的なフィルタで処理速度は速い

カーネル・サイズ（サンプルでは15×15ピクセル）の範囲を平均して全体をぼかし補間する基本的な手法です．単純な計算のため，処理速度は速いもののボケ感も大きいので画質を求める場合には不向きです．処理速度を求める目的以外では使用場面は少ないと思われます．

● CPU版プログラム…Blur.py

▶リスト1：031行…カーネル・サイズの指定

```
ksize = (15, 15)
```

カーネルと呼ばれる（横，縦）のボックス状のデータを表しており，ぼかしを処理するブロック・サイズを指定しています．Ksize変数に配列データを格納していて，数字を大きくするとぼかしの程度も大きくなります．

▶リスト1：033行…対象画像の指定

```
cv2.blur(src=img, ksize=ksize)
```

第1引数：srcは画像です．
第2引数：ksizeはカーネル・サイズです

● CPU版の実行速度

この速度テストはJetson Nanoで実施しています．

```
CPU = 11.50ms
```

GPUはサポートされていないため計測データはありません．

図1の原画に対して**図2**はカーネル15×15，**図3**では55×55で実行した出力結果で，関数の使用方法もシンプルで変換もイメージしやすい出力結果となっています．カーネル（ksize）の値を変更して，ぼかしの程度を確認してみてください．

リスト1　ぼかし（平滑化）プログラムCPU版（Blur.py）

```
004:     cap = cv2.VideoCapture(0, cv2.CAP_V4L)
005:     cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
006:     cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
008:     if not cap.isOpened():
                                # ビデオキャプチャー可能か判断
009:         print("Not Opened Video Camera")
010:         exit()
012:     while True:
013:         ret, img = cap.read()
015:         if not ret:
                        # キャプチャー画像取得に失敗したら終了
016:             print("Video Capture Err")
017:             break
019:         # ここで処理を実行する
020:         img = getBlur(img)
022:         cv2.imshow("Final result", img)
                                                # 画面表示
023:         if cv2.waitKey(10) > -1:
024:             break
026:     cap.release()
027:     cv2.destroyAllWindows()
029: def getBlur(img):
030:     """CPUを使用"""
031:     ksize = (15, 15)
         # 正の奇数で指定する（ここの数字を変えると効果が変更できる）
032:     # フィルターの実行
033:     img = cv2.blur(src=img, ksize=ksize)
```


図1　元画像


図2　ksize=(15, 15)で変換した例


図3　ksize=(55, 55)で変換した例