User's Manual

RZ/A2M グループ

DRP Library ユーザーズマニュアル

ルネサス エレクトロニクス
www.renesas.com

Rev.1.09　2020.12

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1.　本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2.　本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1　2017.11)

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

このマニュアルの使い方

1. 目的と対象者

このマニュアルは「DRP Library」の機能、使用方法をユーザーに理解していただくためのマニュアルです。本ライブラリを用いた応用システムを設計するユーザーを対象にしています。このマニュアルを使用するには、プログラミング言語、マイクロコンピュータに関する基本的な知識が必要です。

本ライブラリは、注意事項を十分確認の上、使用してください。注意事項は、各章の本文中、各章の最後に記載しています。

改訂記録は旧版の記載内容に対して訂正または追加した主な箇所をまとめたものです。改訂内容すべてを記録したものではありません。詳細は、このマニュアルの本文でご確認ください。

目次

1. はじめに

1.1 要旨

本書は RZ/A2M グループのマイクロコンピュータに搭載されている DRP(Dynamically Reconfigurable Processor)上で動作する DRP Library の機能、使い方について説明します。

DRP はユーザーの設定に応じて、様々な機能を実現することができます。本書では、DRP で実現された機能を「回路」と呼び、回路情報を表すデータを「コンフィグレーションデータ」と呼びます。DRP への回路の書き込みは、DRP Driver※でコンフィグレーションデータをロードすることで実現できます。DRP Library は画像処理を中心とした、様々な機能を持つコンフィグレーションデータの集合です。

※ DRP Driver の詳細については、「RZ/A2M グループ DRP Driver ユーザーズマニュアル（R01US0355）」を参照してください。

1.2 機能

DRP Library に含まれるコンフィグレーションデータの機能一覧は、以下に示します。

表 1.1 DRP Library の機能一覧(1/2)

カテゴリ	機能名	概要	ページ
Image processing	Simple ISP	簡易的な ISP をパイプラインで処理します	14
	Simple ISP with object detection by color (HSV)	ターゲットとするオブジェクトの色成分を利用したオブジェクト検出を行う Simple ISP です	22
	Simple ISP with background subtraction	背景差分法により移動物体の抽出を行う Simple ISP です	28
	Simple ISP with object detection using sobel	複数のオブジェクトの中から複雑な輪郭を持つオブジェクトを抽出する Simple ISP です	35
	Simple ISP with distortion correction	樽型歪み補正を行う Simple ISP です	39
	Simple ISP with scaling and normalization(32bit)	AI 推論の前処理（浮動小数点化、正規化、リサイズ）を行う Simple ISP です	45
	Simple ISP with color calibration and 3DNR	カラーマトリクス補正や 3D ノイズリダクションにより、色再現性の高い画像を出力することに特化した Simple ISP です	50
Image transformation	Bayer2Grayscale	CMOS カメラからの RAW データをグレイスケールへ変換します	59
	Bayer2Rgb	CMOS カメラからの RAW データを RGB カラーへ変換します	61
	Bayer2RgbColorCorrection	CMOS カメラからの RAW データを RGB カラーへ変換します(色成分補正有)	67
	Argb2Grayscale	ARGB カラーからグレイスケールへ変換します	70
	BinarizationFixed	画像を固定閾値(Threshold)で二値画像へ変換します	71
	BinarizationAdaptive	画像を周囲画像に合わせた動的閾値で二値画像へ変換します	72
	BinarizationAdaptiveBit	画像を周囲画像に合わせた動的閾値で二値画像へ変換します（ビット出力）	75
	GammaCorrection	画像全体をガンマ値により補正します	77
	Cropping	画像の一部を切り抜きます	78
	CroppingRgb	画像(RGB)の一部を切り抜きます	79
	ResizeBilinearFixed	画像のサイズを変更します(バイリニア法 倍率:2^n 倍)	80
	ResizeBilinearFixedRgb	画像(RGB)のサイズを変更します(バイリニア法 倍率:2^n 倍)	81
	ResizeBilinear	画像のサイズを変更します(バイリニア法 倍率:任意)	82
	ResizeNearest	画像のサイズを変更します(ニアレストネイバー法 倍率:任意)	84
	ImageRotate	画像を回転します	85
	Affine	画像の平行移動、線形変換を行います	88
	Remap	X,Y 座標値マップデータを用いて画像変換を行います	91

表 1.2　DRP Library の機能一覧(2/2)

カテゴリ	機能名	概要	ページ
Image filter	MedianBlur	画像のノイズを除去します (Noise reduction)	96
	GaussianBlur	画像を平滑化します (Smoothing)	97
	UnsharpMasking	画像を鮮鋭化します (Sharpening)	98
	Sobel	Sobel フィルタを使って輪郭を強調した画像を出力します	100
	Prewitt	Prewitt フィルタを使って輪郭を強調した画像を出力します	102
	Laplacian	Laplacian フィルタを使って輪郭を強調した画像を出力します	104
	Dilate	画像の白い部分を膨張させます	106
	Erode	画像の白い部分を収縮させます	108
	Opening ※1	収縮(Erode)のあとに膨張(Dilate)して、黒部分のノイズを除去します	110
	Closing ※1	膨張(Dilate)のあとに収縮(Erode)して、白部分のノイズを除去します	113
Feature detection	CannyCalculate	Canny 法を使って、画像の輪郭を検出します（2 機能の連続処理で実現）	116
	CannyHysterisis		118
	CornerHarris	Chris Harris の考案した手法で画像に含まれる頂点を検出します	120
	MinutiaeExtract	指紋認識で使用される指紋隆線の特徴点を抽出します	122
	MinutiaeDelete	指紋認識で使用される指紋隆線の特徴点を削除します	129
	CircleFitting	円を検出します	139
	FindContours	輪郭を検出し、その外接矩形を算出します	143
Histograms	Histogram	入力画像のヒストグラムを生成します	147
	HistogramNormalization	画像をヒストグラム正規化します	153
	HistogramNormalizationRgb	画像(RGB)をヒストグラム正規化します	156
Other	ReedSolomon	Reed-Solomon 符号を用いた誤り訂正をします（原始多項式固定）	160
	ReedSolomonGf8	GF(2^8)の Reed-Solomon 符号を用いた誤り訂正をします	162
	Thinning	細線化した画像を出力します	164
	ImageMerging	分割して撮影された 2 枚のグレイスケール画像をマージします	168

※1：本機能は Dilate と Erode の組み合わせにより実現します。

2. 動作条件

DRP Library は下記の条件で動作します。

表 2.1 動作条件

項目	内容
マイクロコンピュータ	RZ/A2M グループに属するマイクロコンピュータ※ - R7S921051VCBG - R7S921052VCBG - R7S921053VCBG

※ DRP Library は DRP 機能を搭載した RZ/A2M グループに属するマイクロコンピュータで動作します。DRP 機能を搭載していない RZ/A2M グループに属するマイクロコンピュータでは動作しませんのでご注意ください。

本ライブラリは、以下の環境で動作確認を行いました。

RENESAS e2 studio 7.8.0

対応するツールチェーンは下記となります：

GCC ARM Embedded Toolchain 6-2017-q2-update

3.　ファイル構成

DRP Library のコンフィグレーションデータ、及び、ヘッダファイルのファイル構成を図 3.1、図 3.2 に示します。

```
drp_lib
    r_drp_affine.dat                          Affine
    r_drp_affine.h
    r_drp_argb2grayscale.dat                  ARGB2Grayscale
    r_drp_argb2grayscale.h
    r_drp_bayer2grayscale.dat                 Bayer2Grayscale
    r_drp_bayer2grayscale.h
    r_drp_bayer2rgb.dat                       Bayer2Rgb
    r_drp_bayer2rgb.h
    r_drp_bayer2rgb_color_correction.dat      Bayer2RgbColorCorrection
    r_drp_bayer2rgb_color_correction.h
    r_drp_binarization_adaptive.dat           BinarizationAdaptive
    r_drp_binarization_adaptive.h
    r_drp_binarization_adaptive_bit.dat       BinarizationAdaptiveBit
    r_drp_binarization_adaptive_bit.h
    r_drp_binarization_fixed.dat              BinarizationFixed
    r_drp_binarization_fixed.h
    r_drp_canny_calculate.dat                 CannyCalculate
    r_drp_canny_calculate.h
    r_drp_canny_hysterisis.dat                CannyHysterisis
    r_drp_canny_hysterisis.h
    r_drp_circle_fitting.dat                  CircleFitting
    r_drp_circle_fitting.h
    r_drp_corner_harris.dat                   CornerHarris
    r_drp_corner_harris.h
    r_drp_cropping.dat                        Cropping
    r_drp_cropping.h
    r_drp_cropping_rgb.dat                    CroppingRgb
    r_drp_cropping_rgb.h
    r_drp_dilate.dat                          Dilate
    r_drp_dilate.h
    r_drp_erode.dat                           Erode
    r_drp_erode.h
    r_drp_find_contours.dat                   FindContours
    r_drp_find_contours.h
    r_drp_gamma_correction.dat                GammaCorrection
    r_drp_gamma_correction.h
    r_drp_gaussian_blur.dat                   GaussianBlur
    r_drp_gaussian_blur.h
    r_drp_histogram.dat                       Histogram
    r_drp_histogram.h
    r_drp_histogram_normalization.dat         HistogramNormalization
    r_drp_histogram_normalization.h
    r_drp_histogram_normalization_rgb.dat     HistogramNormalizationRgb
    r_drp_histogram_normalization_rgb.h
    r_drp_image_merging.dat                   ImageMerging
    r_drp_image_merging.h
    r_drp_image_rotate.dat                    ImageRotate
    r_drp_image_rotate.h
```

図 3.1　　ファイル構成(1/2)

r_drp_laplacian.dat	LaplacianFilter
r_drp_laplacian.h	
r_drp_median_blur.dat	MedianBlur
r_drp_median_blur.h	
r_drp_minutiae_delete.dat	MinutiaeDelete
r_drp_minutiae_delete.h	
r_drp_minutiae_extract.dat	MinutiaeExtract
r_drp_minutiae_extract.h	
r_drp_prewitt.dat	Prewitt
r_drp_prewitt.h	
r_drp_reed_solomon.dat	ReedSolomon
r_drp_reed_solomon.h	
r_drp_reed_solomon_gf8.dat	ReedSolomonGf8
r_drp_reed_solomon_gf8.h	
r_drp_remap.dat	Remap
r_drp_remap.h	
r_drp_resize_bilinear.dat	ResizeBilinear
r_drp_resize_bilinear.h	
r_drp_resize_bilinear_fixed.dat	ResizeBilinearFixed
r_drp_resize_bilinear_fixed.h	
r_drp_resize_bilinear_fixed_rgb.dat	ResizeBilinearFixedRgb
r_drp_resize_bilinear_fixed_rgb.h	
r_drp_resize_nearest.dat	ResizeNearest
r_drp_resize_nearest.h	
r_drp_simple_isp_bayer2grayscale_3.dat	Simple ISP
r_drp_simple_isp_bayer2grayscale_6.dat	
r_drp_simple_isp_bayer2rgb_6.dat	
r_drp_simple_isp_bayer2yuv_3.dat	
r_drp_simple_isp_bayer2yuv_6.dat	
r_drp_simple_isp_bayer2yuv_planar_3.dat	
r_drp_simple_isp_bayer2yuv_planar_6.dat	
r_drp_simple_isp_grayscale_3.dat	
r_drp_simple_isp_grayscale_6.dat	
r_drp_simple_isp.h	
r_drp_simple_isp_bg_subtraction_6.dat	Simple ISP with background subtraction
r_drp_simple_isp_bg_subtraction.h	
r_drp_simple_isp_colcal_3dnr_6.dat	Simple ISP with color calibration and 3DNR
r_drp_simple_isp_colcal_3dnr.h	
r_drp_simple_isp_distortion_correction_6.dat	Simple ISP with distortion correction
r_drp_simple_isp_distortion_correction.h	
r_drp_simple_isp_obj_det_color_6.dat	Simple ISP with object detection by color (HSV)
r_drp_simple_isp_obj_det_color.h	
r_drp_simple_isp_obj_det_sobel_4.dat	Simple ISP with object detection using sobel
r_drp_simple_isp_obj_det_sobel_6.dat	
r_drp_simple_isp_obj_det_sobel.h	
r_drp_simple_isp_scal_normaliz_b32_6.dat	Simple ISP with scaling and normalization (32bit)
r_drp_simple_isp_scal_normaliz_b32.h	
r_drp_sobel.dat	Sobel
r_drp_sobel.h	
r_drp_thinning.dat	Thinning
r_drp_thinning.h	
r_drp_unsharp_masking.dat	UnsharpMasking
r_drp_unsharp_masking.h	

図 3.2 ファイル構成(2/2)

4.　DRP Library 仕様

4.1　DRP Library 仕様の読み方

本章では、DRP Library に含まれるコンフィグレーションデータの仕様について、以下の形式で記載しています。

機能名※1	
機能概要	
コンフィグレーションデータファイル	コンフィグレーションデータのファイル名です。DRP Driver の R_DK2_Load()関数で DRP へロードしてください。
対応バージョン	本仕様で動作するコンフィグレーションデータのバージョンを示します。DRP Driver の R_DK2_GetInfo()関数で取得できます。
コンフィグレーションデータサイズ（バイト）	コンフィグレーションデータのサイズを示します。異なるバージョンがある場合はすべてのバージョンについて記載します。
ヘッダファイル	コンフィグレーションデータを使用するためのヘッダファイル名です。#include "ヘッダファイル"でインクルードしてください。
パラメータ	回路で必要とするパラメータを示します。DRP Driver の R_DK2_Start()関数によって、CPU 側から DRP 側にパラメータを渡します。パラメータは、ヘッダファイルの中で構造体として定義されています。回路を実行する前に、CPU 側で各パラメータを設定してください。 stdint.h で定義されているデータタイプを使用します。 また、パラメータを格納した領域、パラメータに指定する src や dst 等のアドレスは、物理メモリ上に存在する必要があり、物理アドレスを設定する必要があります。※2
入出力詳細	パラメータで指定したデータの詳細について示します。入力画像、および出力画像のアドレスについては、特に記載のない限り、同じアドレスを指定可能です。
タイル数	回路が使用するタイル数です。DRP は 6 個のタイルを持っています。回路のタイルへの配置は、DRP Driver の R_DK2_Load()関数を使用します。
分割処理	入力画像を縦方向に分割して、複数の回路で並列に処理可能か示します。 分割処理は、DRP の持つ 6 つのタイルを活用し、3 タイル以下のコンフィグレーションデータを複数ロードすることで実行可能です。3 タイル以下のコンフィグレーションデータを DRP へ複数ロードする方法については、「RZ/A2M グループ DRP Driver ユーザーズマニュアル」の R_DK2_Load()関数の説明を参照してください。 例）入力画像を縦方向に3分割した場合 
解説	コンフィグレーションデータの仕様について説明します。
注意	注意事項があればここに示します。

※1　コンフィグレーションデータの機能名は、DRP Driver の R_DK2_GetInfo()関数でコンフィグレーションデータから取得可能な文字列です。

※2　パラメータを格納する領域、及び、回路の入出力データが Cortex-A9 のキャッシュ上などに存在し、物理メモリの内容がパラメータ、及び、回路の入出力データと一致しない状態になっていると、回路が正常に動作しません。DRP Driver の R_DK2_Start()関数コール前にキャッシュのクリーンを行う、または、パラメータ、及び、回路の入出力データを非キャッシュ領域に配置するなどの対処を行う必要があります。

DRP Driver の API 関数の使用方法については、DRP Driver のユーザーズマニュアル「RZ/A2M グループ DRP Driver ユーザーズマニュアル（R01US0355）」を参照してください。

4.2　Simple ISP

4.2.1　Simple ISP 概要

Simple ISP は画像認識に最適な ISP（Image Signal Processor）であり、メモリ上にあるキャプチャーデータ（ベイヤー配列またはグレイスケール）に対して、色成分補正、色成分積算、デモザイク、ノイズ除去、鮮鋭化、ガンマ補正、カラー変換をパイプライン処理し出力します。出力フォーマット毎に DRP ライブラリを用意しています。入出力フォーマットの一覧と DRP ライブラリのファイル名については、表 4.1 を参照してください。なお、AE（自動露光制御）は、Simple ISP から得た色成分積算値を用い、CPU 側で CMOS センサのゲインやシャッター速度を調整することで実現できます。

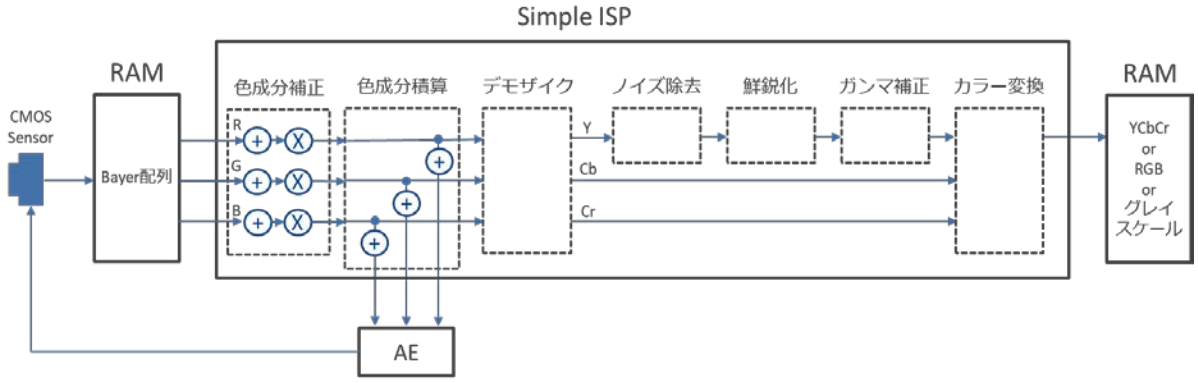


図 4.1　入力がベイヤー配列の場合の Simple ISP ブロック図

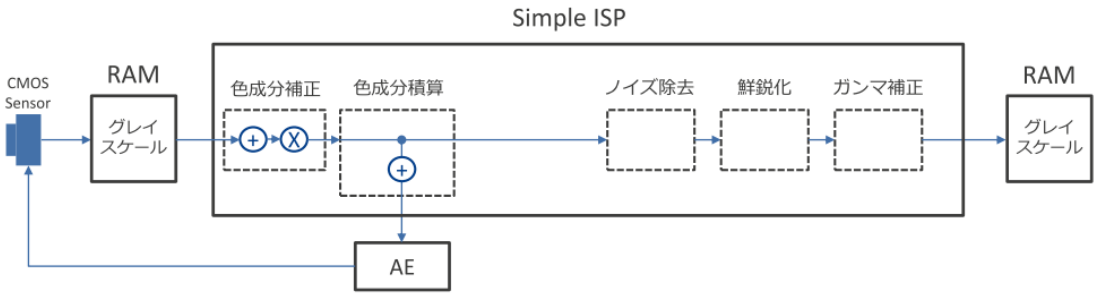


図 4.2　入力がグレイスケールの場合の Simple ISP ブロック図

色成分補正	: ベイヤー配列の各 RGB 成分またはグレイスケールに対して 加算と乗算による補正処理
色成分積算	: ベイヤー配列の各 RGB 成分またはグレイスケールに対しての積算値算出
デモザイク	: ベイヤー配列から YCbCr 成分または Y 成分への補間処理（ACPI/LI）
ノイズ除去	: Y 成分またはグレイスケールに対するノイズ除去処理（Median filter）
鮮鋭化	: Y 成分またはグレイスケールに対する鮮鋭化処理（Unsharp masking）
ガンマ補正	: Y 成分またはグレイスケールに対するガンマ補正処理
カラー変換	: YCbCr 成分に対するカラー変換処理

4.2.2　Simple ISP ライブラリ構成

Simple ISP ライブラリは、下表のように、2 種類の入力と 4 種類の出力フォーマットに対応したコンフィグレーションデータがあります。コンフィグレーションデータには、性能最適化された 6 タイル版と、省タイル構成の 3 タイル版があり、用途に応じて使い分け可能となります。

表 4.1　Simple ISP ライブラリの一覧

入力	出力	タイル数	コンフィグレーションデータファイル名
Bayer	YCbCr Packed format	6 タイル	r_drp_simple_isp_bayer2yuv_6.dat
		3 タイル	r_drp_simple_isp_bayer2yuv_3.dat
	YCbCr Planar format	6 タイル	r_drp_simple_isp_bayer2yuv_planar_6.dat
		3 タイル	r_drp_simple_isp_bayer2yuv_planar_3.dat
	RGB	6 タイル	r_drp_simple_isp_bayer2rgb_6.dat
	グレイスケール	6 タイル	r_drp_simple_isp_bayer2grayscale_6.dat
		3 タイル	r_drp_simple_isp_bayer2grayscale_3.dat
グレイスケール	グレイスケール	6 タイル	r_drp_simple_isp_grayscale_6.dat
		3 タイル	r_drp_simple_isp_grayscale_3.dat

4.2.3 Simple ISP API

Simple ISP

簡易的な ISP をパイプラインで処理します

コンフィグレーションデータファイル	1) r_drp_simple_isp_bayer2yuv_6.dat 2) r_drp_simple_isp_bayer2yuv_3.dat 3) r_drp_simple_isp_bayer2yuv_planar_6.dat 4) r_drp_simple_isp_bayer2yuv_planar_3.dat 5) r_drp_simple_isp_bayer2rgb_6.dat 6) r_drp_simple_isp_bayer2grayscale_6.dat 7) r_drp_simple_isp_bayer2grayscale_3.dat 8) r_drp_simple_isp_grayscale_6.dat 9) r_drp_simple_isp_grayscale_3.dat
対応バージョン	1.02
コンフィグレーションデータサイズ（バイト）	1) 424960、2) 234656、3) 547936、4) 266304、 5) 412128、6) 369344、7) 200512、8) 353568、 9) 205152、
ヘッダファイル	r_drp_simple_isp.h

パラメータ　構造体名

r_drp_simple_isp_t

メンバ名	型	説明
src	uint32_t	入力画像のアドレス
dst	uint32_t	出力画像のアドレス 3)、4)のコンフィグレーションデータでは Y 成分を出力
dst2	uint32_t	出力画像のアドレス（Cb 成分）
dst3	uint32_t	出力画像のアドレス（Cr 成分）
width	uint16_t	画像の幅（最小値 16、最大値は表 4.2 に記載、2 の整数倍）
height	uint16_t	画像の高さ（4～1080、2 の整数倍）
component	uint8_t	1：色成分積算を取得する、0：色成分積算を取得しない
accumulate	uint32_t	色成分積算値の格納先のアドレス
area1_offset_x	uint16_t	色成分積算する領域 1 の開始位置の X 座標
area1_offset_y	uint16_t	色成分積算する領域 1 の開始位置の Y 座標
area1_width	uint16_t	色成分積算する領域 1 の幅
area1_height	uint16_t	色成分積算する領域 1 の高さ
area2_offset_x	uint16_t	色成分積算する領域 2 の開始位置の X 座標
area2_offset_y	uint16_t	色成分積算する領域 2 の開始位置の Y 座標
area2_width	uint16_t	色成分積算する領域 2 の幅
area2_height	uint16_t	色成分積算する領域 2 の高さ
area3_offset_x	uint16_t	色成分積算する領域 3 の開始位置の X 座標
area3_offset_y	uint16_t	色成分積算する領域 3 の開始位置の Y 座標
area3_width	uint16_t	色成分積算する領域 3 の幅
area3_height	uint16_t	色成分積算する領域 3 の高さ
bias_r	int8_t	画像のバイアス補正値（R 成分）（-128～127）
bias_g	int8_t	画像のバイアス補正値（G 成分）（-128～127）
bias_b	int8_t	画像のバイアス補正値（B 成分）（-128～127）
bias_gray	int8_t	画像のバイアス補正値（グレイスケール）（-128～127）
gain_r	uint16_t	画像のゲイン補正値（R 成分） 上位 4bit が整数部、下位 12bit が小数部
gain_g	uint16_t	画像のゲイン補正値（G 成分） 上位 4bit が整数部、下位 12bit が小数部
gain_b	uint16_t	画像のゲイン補正値（B 成分） 上位 4bit が整数部、下位 12bit が小数部

	gain_gray	uint16_t	画像のゲイン補正値（グレイスケール） 上位 4bit が整数部、下位 12bit が小数部
	blend	uint16_t	ノイズ除去の強度（0x000～0x100） 0x000：OFF、0x100：ON（最大）
	strength	uint8_t	鮮鋭化フィルタの強調度の値（0～255）
	coring	uint8_t	鮮鋭化フィルタのコアリングの値（0～255）
	gamma	uint8_t	1：ガンマ補正あり、0：ガンマ補正なし
	table	uint32_t	ガンマ補正に使う LUT アドレス
タイル数	表 4.2 に記載		
分割処理	不可		

表 4.2　Simple ISP のパラメータ及びタイル数対応表

パラメータ	コンフィグレーションデータファイル								
	1)	2)	3)	4)	5)	6)	7)	8)	9)
dst2	×	×	○	○	×	×	×	×	×
dst3	×	×	○	○	×	×	×	×	×
width最大値	1920	1022	1920	1022	1920	1920	1920	1920	1920
bias_r	○	○	○	○	○	○	○	×	×
bias_g	○	○	○	○	○	○	○	×	×
bias_b	○	○	○	○	○	○	○	×	×
bias_gray	×	×	×	×	×	×	×	○	○
gain_r	○	○	○	○	○	○	○	×	×
gain_g	○	○	○	○	○	○	○	×	×
gain_b	○	○	○	○	○	○	○	×	×
gain_gray	×	×	×	×	×	×	×	○	○
タイル数	6	3	6	3	6	6	3	6	3

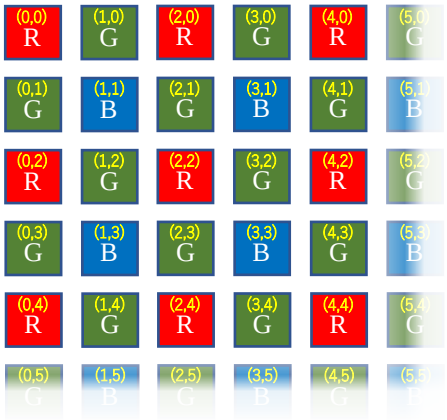
○：使用、×：未使用

上記以外のパラメータは、すべてのコンフィグレーションデータファイルで使用します。

解説

入力画像

入力画像のベイヤー配列を下図に示します。1 ピクセルのデータ長は 8bit としてください。



出力画像

出力が YCbCr422 で Packed format の場合は、YCbCr422（16BPP）データが、パラメータ width、height で指定した画像サイズで出力されます。

YCbCr422 Packed format の出力フォーマット：
CB0,Y0,CR0,Y1,CB2,Y2,CR2,Y3・・・・・
（Yn, CBn, CRn: n 番目のピクセルの輝度と色差の値）

出力が YCbCr422 で Planar format の場合は、YCbCr422（16BPP）データに対し、パラメータ width、height で指定した画像サイズで、dst に Y 成分、dst2 に Cb 成分、dst3 に Cr 成分が出力されます。

YCbCr422 Planar format の dst の出力フォーマット：
Y0,Y1,Y2,Y3・・・・・
YCbCr422 Planar format の dst2 の出力フォーマット：
CB0,CB2,CB4,CB6・・・・・
YCbCr422 Planar format の dst3 の出力フォーマット：
CR0,CR2,CR4,CR6・・・・・
（Yn, CBn, CRn: n 番目のピクセルの輝度と色差の値）

出力が RGB の場合は、RGB（24BPP）データが、パラメータ width、height で指定した画像サイズで出力されます。

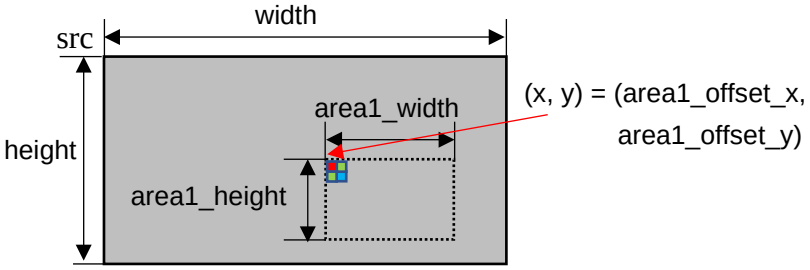
RGB の出力フォーマット：
R0,G0,B0,R1,G1,B1・・・・・
（Rn, Gn, Bn: n 番目のピクセルの各色の明度）

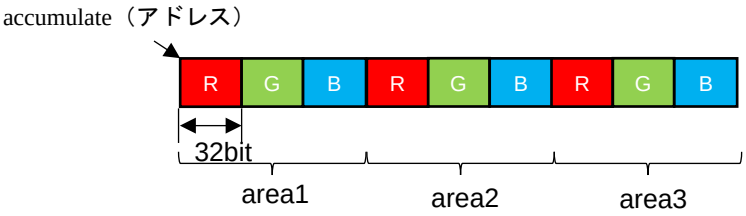
出力がグレイスケールの場合は、グレイスケール（8BPP）データが、パラメータ width、height で指定した画像サイズで出力されます。

各パイプライン処理詳細

■色成分積算

ベイヤー配列の RGB 各成分に対し積算した結果を出力します。パラメータの area1 から area3 で指定した 3 つの領域に対し、R 成分、G 成分、B 成分の 3 つの成分毎に積算します。グレイスケールの場合はすべての結果に同じ積算値を出力します。合計 9 つの積算値が、accumulate で指定されたアドレスへ出力されます。1 積算値=32bit 長となるので、合計 36 バイト分の領域を確保してください。





色成分積算値から入力がベイヤー配列の場合の平均輝度は以下の式で算出できます。

$$\text{平均輝度} = \frac{(0.299 \times \text{R の積算} \times 4) + (0.587 \times \text{G の積算} \times 2) + (0.114 \times \text{B の積算} \times 4)}{\text{領域の幅} \times \text{領域の高さ}}$$

また、色成分の平均は以下の式で算出できます。

$$\text{色成分の平均（R or B）} = \frac{\text{R or B の積算}}{\text{領域の幅} \times \text{領域の高さ} \div 4}$$

$$\text{色成分の平均（G）} = \frac{\text{G の積算}}{\text{領域の幅} \times \text{領域の高さ} \div 2}$$

■色成分補正

入力がベイヤー配列の場合、RGB の各成分に対し、パラメータ bias_r、bias_g、bias_b で設定した値が加算され、gain_r、gain_g、gain_b で設定した値が乗算されます。

入力がグレイスケールの場合、パラメータ bias_gray で設定した値が加算され、gain_gray で設定した値が乗算されます。

■デモザイク

YCbCr 出力と RGB 出力では、適応型カラープレーン補間法（ACPI）によりベイヤー配列から YCbCr422 へ変換します。

グレイスケール出力では、線形補間法（LI）により、ベイヤー配列からグレイスケールへ変換します。

■ノイズ除去

Median filter アルゴリズムによりノイズ除去を行います。
入力画像と Median filter ノイズ除去後の画像を、パラメータ blend の割合で合成することで、ノイズ除去する量を調整できます。blend に 0 を指定した場合は、ノイズ除去が OFF になります。

$$出力 = \frac{入力画像 \times (256 - \mathrm{blend}) + \mathrm{median}\ 画像 \times \mathrm{blend}}{256}$$

■鮮鋭化

Unsharp masking アルゴリズムにより画像の鮮鋭化を行います。入力に対して、以下の 8 方向ラプラシアンフィルタで作成したエッジを減算することで鮮鋭化を行います。鮮鋭化の強度を strength、鮮鋭化を行わない振幅差のしきい値を coring で指定します。

8 方向ラプラシアンフィルタ

1	1	1
1	-8	1
1	1	1

鮮鋭化処理計算は以下のとおりです。

$$出力 = 入力 - \left(\frac{\mathrm{strength}}{256} \times \mathrm{A}\right)$$

A：8 方向ラプラシアンフィルタをかけた結果

エッジが弱い低振幅差に対して鮮鋭化処理を実行しないように coring で比較します。以下の式を満たしている注目画素はフィルタ処理を行いません。

$$\mathrm{coring} \geq |\mathrm{A}|$$

■ガンマ補正

パラメータ table で指定したアドレスにガンマテーブルを格納してください。0～255 の table で指定された LUT を参照して画素値の変換を行います。

■カラー変換

出力に合わせてカラー変換を行います。

使用例

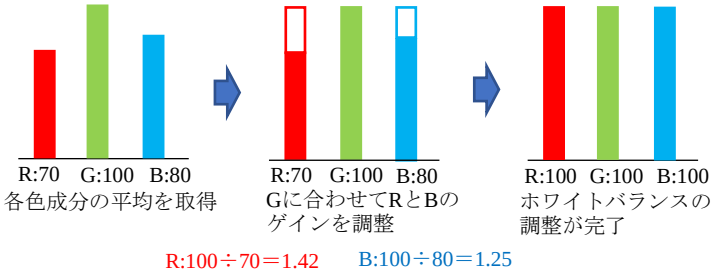
■露光制御の例

色成分積算の結果から平均輝度を算出し、平均輝度を用いて、露光制御を行うことができます。平均輝度が低い場合は、シャッタースピードを遅くする、ゲインを上げる、平均輝度が高い場合は、シャッタースピードを速くする、ゲインを下げることで対応できます。

■ホワイトバランスの例

色成分積算の結果を用いて、ゲイン補正を以下のように行うことでホワイトバランスの調整を行うことができます。G 成分を主体として、R、B の色成分の積算結果を比較し、その比率からゲインの設定値を計算します。

例）



上記例の場合は、R より G が 1.42 倍、B より G が 1.25 倍なので、R のゲインを 1.42 倍、B のゲインを 1.25 倍と設定してください。

注意	なし

4.3　Simple ISP with object detection by color (HSV)

4.3.1　概要

本機能は、メモリ上にあるキャプチャーデータ（ベイヤー配列）に対して、オブジェクト検出の前処理として色成分補正、色成分積算、デモザイク、二値化、Opening、Closing をパイプライン処理します。本機能はターゲットとしているオブジェクトの色成分を利用したオブジェクト検出を対象としています。対象とする色成分を抽出した二値画像及び、キャプチャーデータをグレイスケール化した画像を出力します。なお、AE（自動露光制御）は、本機能から得た色成分積算値を用い、CPU 側で CMOS センサのゲインやシャッター速度を調整することで実現できます。

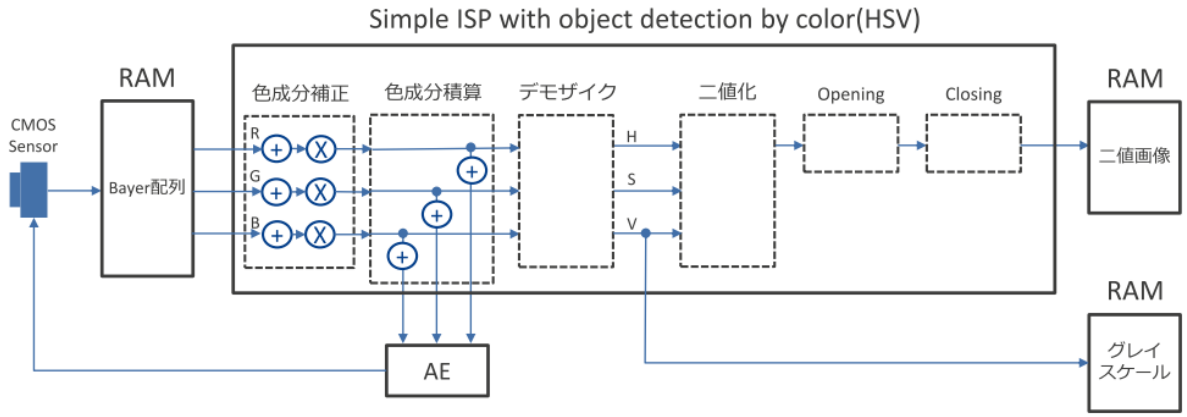


図 4.3　Simple ISP with object detection by color (HSV)ブロック図

色成分補正	: ベイヤー配列の各 RGB 成分に対して加算と乗算による補正処理
色成分積算	: ベイヤー配列の各 RGB 成分に対しての積算値算出
デモザイク	: ベイヤー配列から HSV 成分への補間処理（LI）
二値化	: HSV 成分に対する二値化処理
Opening	: 二値画像に対するノイズ除去処理
Closing	: 二値画像に対するノイズ除去処理

4.3.2 API

Simple ISP with object detection by color (HSV)

ターゲットとするオブジェクトの色成分を利用したオブジェクト検出を行う Simple ISP です

コンフィグレーションデータファイル	r_drp_simple_isp_obj_det_color_6.dat
対応バージョン	1.00
コンフィグレーションデータサイズ（バイト）	322624
ヘッダファイル	r_drp_simple_isp_obj_det_color.h

パラメータ　構造体名

r_drp_simple_isp_obj_det_color_t

メンバ名	型	説明
src	uint32_t	入力画像のアドレス
dst	uint32_t	出力画像（二値画像）のアドレス
v_dst	uint32_t	出力画像（グレイスケール画像）のアドレス （0 を指定した場合は、グレイスケール画像は出力されません）
width	uint16_t	画像の幅（16～1920、2 の整数倍）
height	uint16_t	画像の高さ（4～1080、2 の整数倍）
component	uint8_t	1：色成分積算を取得する、0：色成分積算を取得しない
accumulate	uint32_t	色成分積算値の格納先のアドレス
area1_offset_x	uint16_t	色成分積算する領域 1 の開始位置の X 座標
area1_offset_y	uint16_t	色成分積算する領域 1 の開始位置の Y 座標
area1_width	uint16_t	色成分積算する領域 1 の幅
area1_height	uint16_t	色成分積算する領域 1 の高さ
area2_offset_x	uint16_t	色成分積算する領域 2 の開始位置の X 座標
area2_offset_y	uint16_t	色成分積算する領域 2 の開始位置の Y 座標
area2_width	uint16_t	色成分積算する領域 2 の幅
area2_height	uint16_t	色成分積算する領域 2 の高さ
area3_offset_x	uint16_t	色成分積算する領域 3 の開始位置の X 座標
area3_offset_y	uint16_t	色成分積算する領域 3 の開始位置の Y 座標
area3_width	uint16_t	色成分積算する領域 3 の幅
area3_height	uint16_t	色成分積算する領域 3 の高さ
bias_r	int8_t	画像のバイアス補正値（R 成分）（-128～127）
bias_g	int8_t	画像のバイアス補正値（G 成分）（-128～127）
bias_b	int8_t	画像のバイアス補正値（B 成分）（-128～127）
gain_r	uint16_t	画像のゲイン補正値（R 成分） 上位 4bit が整数部、下位 12bit が小数部
gain_g	uint16_t	画像のゲイン補正値（G 成分） 上位 4bit が整数部、下位 12bit が小数部
gain_b	uint16_t	画像のゲイン補正値（B 成分） 上位 4bit が整数部、下位 12bit が小数部
h_cmp_mode	uint8_t	0:二値化の判定に使用する H 成分の範囲は 0 を跨がない 1:二値化の判定に使用する H 成分の範囲は 0 を跨ぐ
h_min	uint8_t	二値化における下限判定値（H 成分） （h_cmp_mode＝0 の場合 0～h_max、 h_cmp_mode＝1 の場合 h_max～255）
h_max	uint8_t	二値化における上限判定値（H 成分） （h_cmp_mode＝0 の場合 h_min～255、 h_cmp_mode＝1 の場合 0～h_min）
s_min	uint8_t	二値化における下限判定値（S 成分）（0～s_max）
s_max	uint8_t	二値化における上限判定値（S 成分）（s_min～255）

	v_min	uint8_t	二値化における下限判定値（V 成分）（0～v_max）
	v_max	uint8_t	二値化における上限判定値（V 成分）（v_min～255）
	opening	uint8_t	Opening 処理の Erode と Dilate の繰り返し回数 （0～30、2 の整数倍）
	closing	uint8_t	Closing 処理の Dilate と Erode の繰り返し回数 （0～30、2 の整数倍）
タイル数	6		
分割処理	不可		

解説

入力画像

「4.2.3 Simple ISP API」の入力画像と同じです。

出力画像

dst 領域に、二値化（8BPP）データが、パラメータ width、height で指定した画像サイズで出力されます。

v_dst 領域に、グレイスケール（8BPP）データが、パラメータ width、height で指定した画像サイズで出力されます。

各パイプライン処理詳細

■色成分補正

「4.2.3 Simple ISP API」の色成分補正と同じです。

■色成分積算

「4.2.3 Simple ISP API」の色成分積算と同じです。

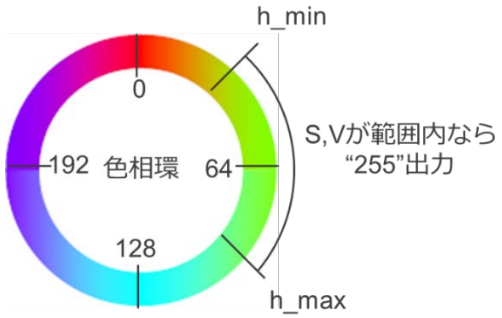
■デモザイク

線形補間法（LI）により、ベイヤー配列から HSV へ変換します。線形補間法は偽色が発生する可能性があり、発生した偽色は H 成分の閾値を元に二値化を行うと細かな点状のノイズとなる可能性があります。しかし、パラメータ dst に出力される二値画像は、Opening、Closing 処理で細かな点状のノイズを除去することが可能であり、パラメータ v_dst に出力されるグレイスケール画像は、色成分を含まないため影響が小さいので、処理の軽い線形補間法を使用しています。

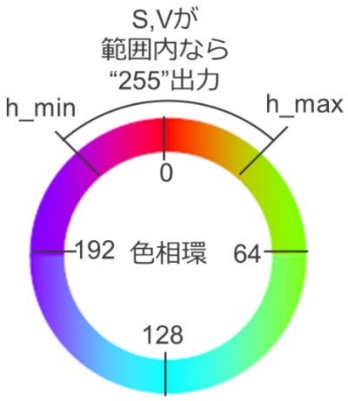
■二値化

HSV データの各成分が、以下の 1～3 の条件を全て満たす場合は 255、それ以外は 0 に二値化します。

1. H 成分が、下記の条件を満たす
   A) h_cmp_mode=0 の場合
      「h_min ≦ H 成分 ≦ h_max」



   B) h_cmp_mode=1 の場合
      「h_min ≦ H 成分」or「H 成分 ≦ h_max」



2. s_min ≦ S 成分 ≦ s_max
3. v_min ≦ V 成分 ≦ v_max

■Opening

繰り返し回数が opening パラメータ回の Opening 処理を行います。Opening 処理の詳細については、4.10.9 Opening を参照してください。

■Closing

繰り返し回数が closing パラメータ回の Closing 処理を行います。Closing 処理の詳細については、4.10.10 Closing を参照してください。

使用例

■色検出の例

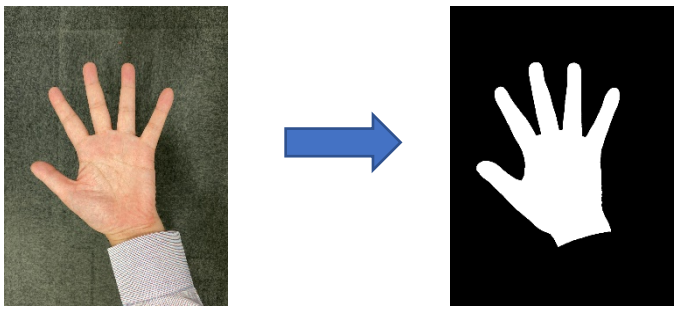
本機能では、色成分の抽出を目的としているため、色成分抽出に適した色空間である HSV 色空間に変換しています。

HSV の各成分に対して、閾値を指定して二値化することで、特定の色成分の抽出が可能です。
例として、人の手を検出する場合は、手の色を抽出するように各パラメータを設定します。
ただし、環境によってパラメータ h_cmp_mode、*_min、*_max(*=h,s,v)の設定値は調整してください。

パラメータ設定例：

```
h_cmp_mode=0
h_min=0, h_max=14
s_min=30, s_max=150
v_min=60, v_max=255
```

入力画像　　出力画像(dst)



■露光制御の例

「4.2.3 Simple ISP API」の露光制御の例と同じです。

■ホワイトバランスの例

「4.2.3 Simple ISP API」のホワイトバランスの例と同じです。

注意　なし

4.4　Simple ISP with background subtraction

4.4.1　概要

本機能は背景差分法により入力画像と背景モデル画像を比較し、差分として抽出した移動物体の二値画像を出力することに特化した Simple ISP です。これを用いることによってユーザーは CMOS センサの入力から移動物体を抽出した二値画像を得るまでを本機能一つで実現できます。本機能は、メモリ上にあるキャプチャーデータ（ベイヤー配列）に対して、色成分補正、色成分積算、デモザイク、ノイズ除去、鮮鋭化、ガンマ補正、背景差分をパイプライン処理します。なお、AE（自動露光制御）は、本機能から得た色成分積算値を用い、CPU 側で CMOS センサのゲインやシャッター速度を調整することで実現できます。

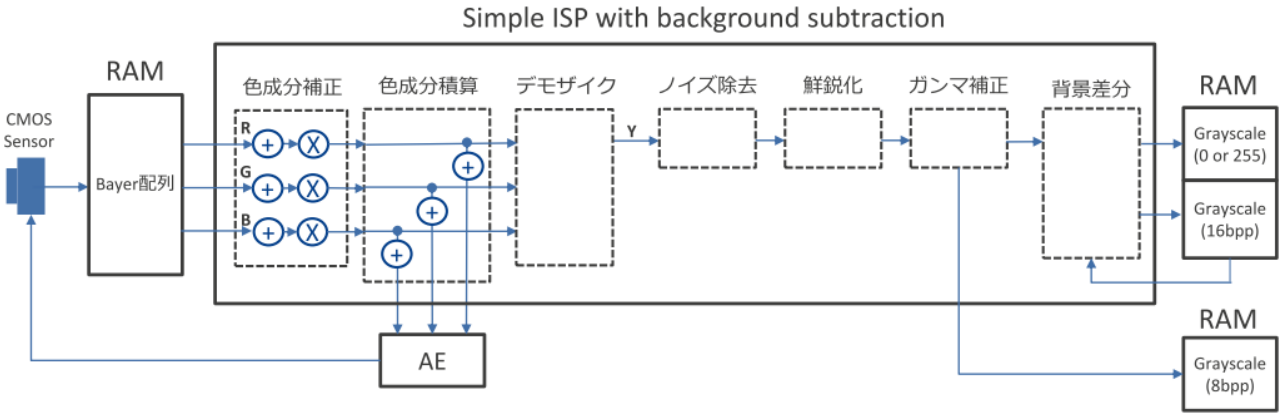


図 4.4　Simple ISP with background subtraction ブロック図

色成分補正	: ベイヤー配列の各 RGB 成分に対して加算と乗算による補正処理
色成分積算	: ベイヤー配列の各 RGB 成分に対しての積算値算出
デモザイク	: ベイヤー配列から Y 成分への補間処理（LI）
ノイズ除去	: Y 成分に対するノイズ除去処理（Median filter）
鮮鋭化	: Y 成分に対する鮮鋭化処理（Unsharp masking）
ガンマ補正	: Y 成分に対するガンマ補正処理
背景差分	: 背景差分法による移動物体の抽出処理

4.4.2 API

Simple ISP with background subtraction

背景差分法により移動物体の抽出を行う Simple ISP です

コンフィグレーションデータファイル	r_drp_simple_isp_bg_subtraction_6.dat
対応バージョン	1.00
コンフィグレーションデータサイズ（バイト）	450880
ヘッダファイル	r_drp_simple_isp_bg_subtraction.h

パラメータ　構造体名

r_drp_simple_isp_bg_subtraction_t

メンバ名	型	説明
src	uint32_t	入力画像のアドレス
dst1	uint32_t	移動物体抽出画像の出力アドレス
dst2	uint32_t	背景モデル画像の入出力アドレス
dst3	uint32_t	出力画像（グレイスケール画像）のアドレス （0 を指定した場合は、グレイスケール画像は出力されません）
width	uint16_t	画像の幅（16～1920、4 の整数倍）
height	uint16_t	画像の高さ（4～1080、2 の整数倍）
component	uint8_t	1：色成分積算を取得する、0：色成分積算を取得しない
accumulate	uint32_t	色成分積算値の格納先のアドレス
area1_offset_x	uint16_t	色成分積算する領域 1 の開始位置の X 座標
area1_offset_y	uint16_t	色成分積算する領域 1 の開始位置の Y 座標
area1_width	uint16_t	色成分積算する領域 1 の幅
area1_height	uint16_t	色成分積算する領域 1 の高さ
area2_offset_x	uint16_t	色成分積算する領域 2 の開始位置の X 座標
area2_offset_y	uint16_t	色成分積算する領域 2 の開始位置の Y 座標
area2_width	uint16_t	色成分積算する領域 2 の幅
area2_height	uint16_t	色成分積算する領域 2 の高さ
area3_offset_x	uint16_t	色成分積算する領域 3 の開始位置の X 座標
area3_offset_y	uint16_t	色成分積算する領域 3 の開始位置の Y 座標
area3_width	uint16_t	色成分積算する領域 3 の幅
area3_height	uint16_t	色成分積算する領域 3 の高さ
bias_r	int8_t	画像のバイアス補正値（R 成分）（-128～127）
bias_g	int8_t	画像のバイアス補正値（G 成分）（-128～127）
bias_b	int8_t	画像のバイアス補正値（B 成分）（-128～127）
gain_r	uint16_t	画像のゲイン補正値（R 成分） 上位 4bit が整数部、下位 12bit が小数部
gain_g	uint16_t	画像のゲイン補正値（G 成分） 上位 4bit が整数部、下位 12bit が小数部
gain_b	uint16_t	画像のゲイン補正値（B 成分） 上位 4bit が整数部、下位 12bit が小数部
blend	uint16_t	ノイズ除去の強度（0x000～0x100） 0x000：OFF、0x100：ON（最大）
strength	uint8_t	鮮鋭化フィルタの強調度の値（0～255）
coring	uint8_t	鮮鋭化フィルタのコアリングの値（0～255）
gamma	uint8_t	1：ガンマ補正あり、0：ガンマ補正なし
table	uint32_t	ガンマ補正に使う LUT アドレス
mean_img_init	uint8_t	1：背景モデルを初期化する、0：背景モデルを初期化しない

	alpha	uint8_t	背景モデル更新時の入力画像の重み（0～255） 検出精度を調整します。alpha の値を大きくすると、検出精度が低くなりますが、誤検出の可能性が低くなります。alpha の値を小さくすると、検出精度が高くなりますが、誤検出の可能性が高くなります。 詳細は使用例を参照してください。
	threshold_latest	uint8_t	入力画像の二値化の閾値（0～255）
	threshold_diff	uint8_t	入力画像と背景モデルの差分画像の二値化の閾値（0～255）
タイル数	6		
分割処理	不可		

解説

入力画像

「4.2.3 Simple ISP API」の入力画像と同じです。

出力画像

dst1 領域に、抽出した移動物体が二値化画像で出力されます。移動物体が存在する領域のピクセル値を 255、それ以外を 0 とし、width と height で指定した画像サイズで出力されます。dst1 は src と同一アドレスを指定することが可能です。

dst2 領域に、背景モデル画像（16bpp グレイスケール）が、width と height で指定した画像サイズで出力されます。

dst3 領域に、ガンマ補正後のグレイスケール（8BPP）データが、パラメータ width、height で指定した画像サイズで出力されます。

各パイプライン処理詳細

■色成分積算

「4.2.3 Simple ISP API」の色成分積算と同じです。

■色成分補正

「4.2.3 Simple ISP API」の色成分補正と同じです。

■デモザイク

線形補間法（LI）により、ベイヤー配列からグレイスケールへ変換します。

■ノイズ除去

「4.2.3 Simple ISP API」のノイズ除去と同じです。

■鮮鋭化

「4.2.3 Simple ISP API」の鮮鋭化と同じです。

■ガンマ補正

「4.2.3 Simple ISP API」のガンマ補正と同じです。

■背景差分

ガンマ補正後の入力画像と背景モデル画像（過去画像の加重移動平均）を比較することで移動物体抽出画像を出力します。背景差分の処理内容を以下に示します。

①dst2 から過去画像の背景モデルを読み込み、過去画像の背景モデルと入力画像より背景モデル画像の更新を行い、dst2 へ出力します。

・mean_img_init=1 の場合

更新背景モデル画像 ＝入力画像

・mean_img_init=0 の場合

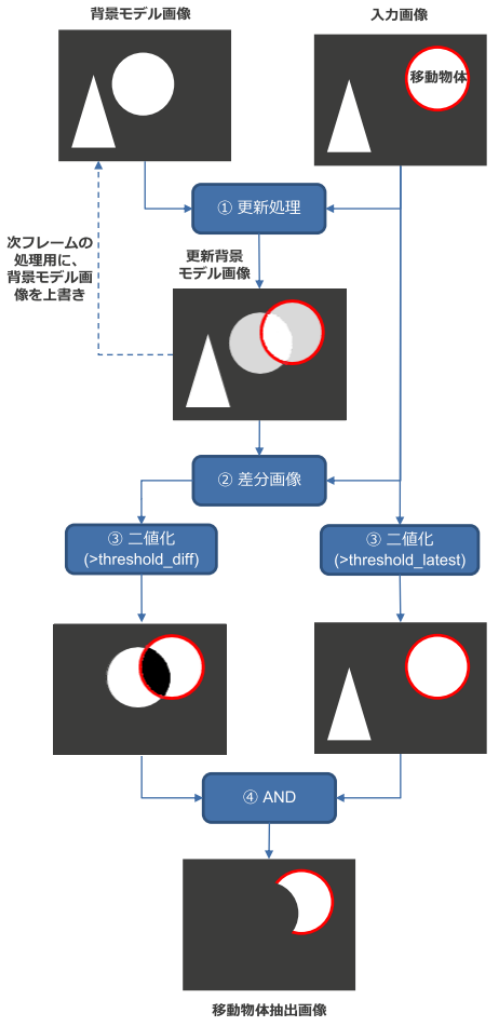
更新背景モデル画像 $= \left((256 - alpha) \times (\text{過去画像の背景モデル}) + alpha \times (\text{入力画像})\right) \div 256$

②入力画像と背景モデルの差分画像を以下の計算により求めます。

$(\text{差分画像}) = |(\text{入力画像}) - (\text{更新背景モデル画像})|$

③入力画像を閾値（threshold_latest）で二値化します。差分画像を閾値（threshold_diff）で二値化します。入力データが閾値を超えた場合は 255、閾値以下の場合は 0 を出力します。

④二値化した入力画像と差分画像の論理積を移動物体抽出画像として dst1 へ出力します。



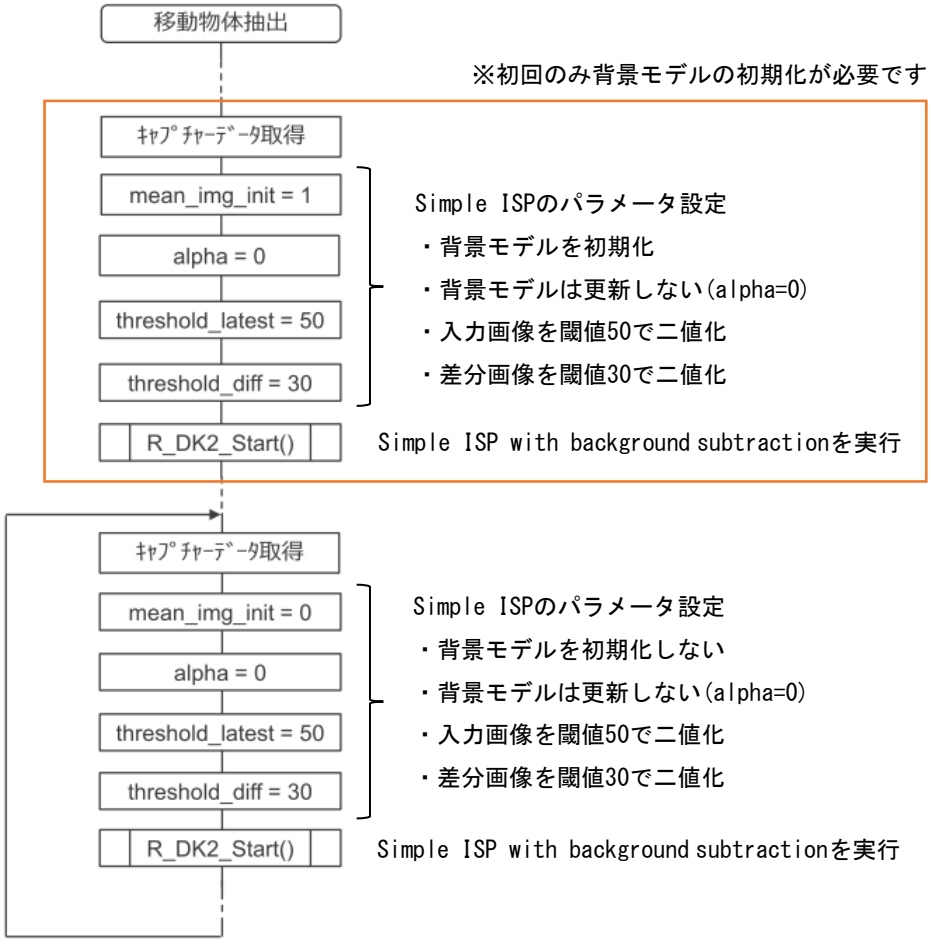
使用例

■動作フロー

本機能では、背景モデル画像を任意に設定し、入力画像と背景モデル画像の差分画像を求め、二値化した入力画像と差分画像の論理積をとることで、固定された背景の中の移動物体を検出することが可能です。加えて aplha の値を調整することで、微小な背景の変化にも対応可能です。背景が固定されている場合と背景が変化する場合のフローを以下に示します。

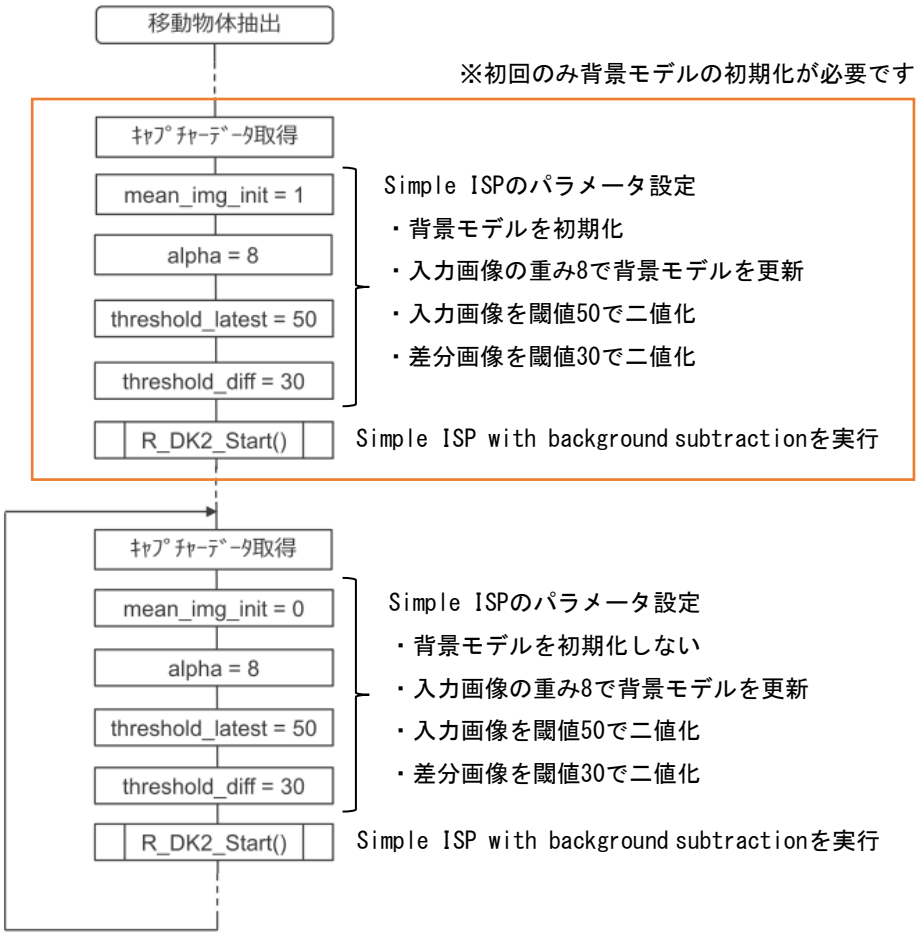
[背景が固定されている場合]
固定の画像を背景モデルとして設定し、背景モデルの更新は行わないことで、移動物体を抽出できます。alpha の値は 0 を設定します。

パラメータ設定例：



[背景が固定されない場合]
毎フレーム毎に入力画像で背景モデル画像の更新を行うことで、移動物体を抽出できます。
alpha の値は 0 以外の値を設定します。例としてカメラで移動する物体を検出する場合、alpha の値は移動物体とカメラの移動速度から決定する必要があります。
alpha の値を小さくすると、移動物体の小さな動きでも検出されやすくなります。また、移動体が停止した後の検出し続ける時間は長くなります。ただし、カメラが小さく動いたとしても背景を移動物体として誤検出する可能性が高くなります。
alpha の値を大きくすると、移動物体の小さな動きは検出されにくくなります。また、移動体が停止した後の検出し続ける時間は短くなります。ただし、カメラの動きによる背景を移動物体として誤検出する可能性は低くなります。

パラメータ設定例：



■露光制御の例

「4.2.3 Simple ISP API」の露光制御の例と同内容になります。

■ホワイトバランスの例

「4.2.3 Simple ISP API」のホワイトバランスの例と同内容になります。

注意	なし

4.5　Simple ISP with object detection using sobel

4.5.1　概要

本機能は、複数のオブジェクトの中から 2D Barcode などの複雑な輪郭を持つオブジェクトを抽出する Simple ISP です。Sobel フィルタによる輪郭の抽出、Dilate による輪郭の強調、Erode による不要な輪郭の削除により、複雑な輪郭を持つオブジェクトの二値画像を出力します。本章の解説の中で 2D Barcode デコードの前処理を行う例を説明します。本機能は、メモリ上にあるキャプチャーデータ（ベイヤー配列）に対して、色成分補正、色成分積算、デモザイク、ノイズ除去、Sobel、二値化、Dilate、Erode をパイプライン処理します。また、エッジ成分を抽出した二値画像と、キャプチャーデータをグレイスケール化した画像を出力します。なお、AE（自動露光制御）は、本機能から得た色成分積算値を用い、CPU 側で CMOS センサのゲインやシャッター速度を調整することで実現できます。

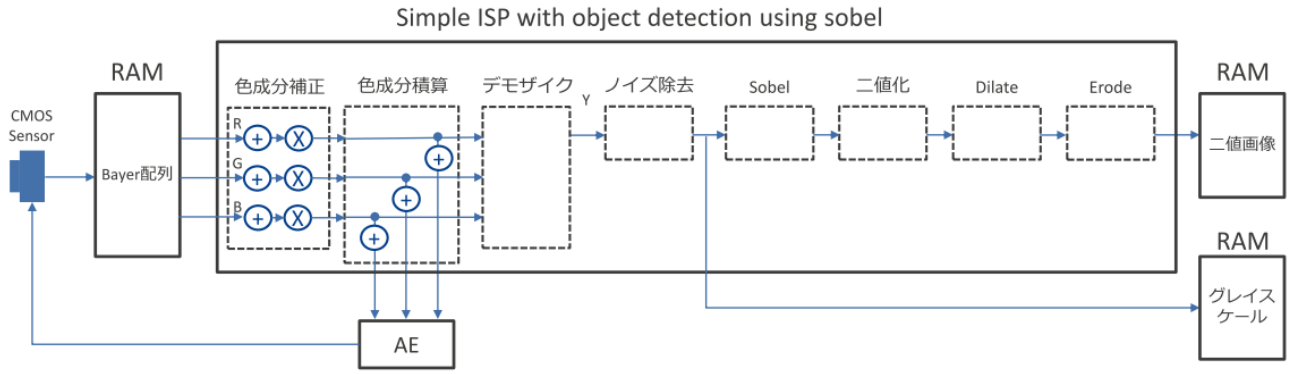


図 4.5　Simple ISP with object detection using sobel ブロック図

色成分補正	: ベイヤー配列の各 RGB 成分に対して加算と乗算による補正処理
色成分積算	: ベイヤー配列の各 RGB 成分に対しての積算値算出
デモザイク	: ベイヤー配列から Y 成分への補間処理（LI）
ノイズ除去	: Y 成分に対するノイズ除去処理（Median filter）
Sobel	: 輪郭抽出処理
二値化	: 二値化処理
Dilate	: 二値画像に対する白部分膨張処理
Erode	: 二値画像に対する白部分収縮処理

4.5.2 API

Simple ISP with object detection using sobel

複数のオブジェクトの中から複雑な輪郭を持つオブジェクトを抽出する Simple ISP です

コンフィグレーションデータファイル		1) r_drp_simple_isp_obj_det_sobel_6.dat 2) r_drp_simple_isp_obj_det_sobel_4.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		1) 350016、2) 282944	
ヘッダファイル		r_drp_simple_isp_obj_det_sobel.h	
パラメータ	構造体名		
	r_drp_simple_isp_obj_det_sobel_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst1	uint32_t	出力画像（二値画像）のアドレス
	dst2	uint32_t	出力画像（グレイスケール画像）のアドレス （0 を指定した場合は、グレイスケール画像は出力されません）
	width	uint16_t	画像の幅（16～1920、2 の整数倍）
	height	uint16_t	画像の高さ（4～1080、2 の整数倍）
	component	uint8_t	1：色成分積算を取得する、0：色成分積算を取得しない
	accumulate	uint32_t	色成分積算値の格納先のアドレス
	area1_offset_x	uint16_t	色成分積算する領域 1 の開始位置の X 座標
	area1_offset_y	uint16_t	色成分積算する領域 1 の開始位置の Y 座標
	area1_width	uint16_t	色成分積算する領域 1 の幅
	area1_height	uint16_t	色成分積算する領域 1 の高さ
	area2_offset_x	uint16_t	色成分積算する領域 2 の開始位置の X 座標
	area2_offset_y	uint16_t	色成分積算する領域 2 の開始位置の Y 座標
	area2_width	uint16_t	色成分積算する領域 2 の幅
	area2_height	uint16_t	色成分積算する領域 2 の高さ
	area3_offset_x	uint16_t	色成分積算する領域 3 の開始位置の X 座標
	area3_offset_y	uint16_t	色成分積算する領域 3 の開始位置の Y 座標
	area3_width	uint16_t	色成分積算する領域 3 の幅
	area3_height	uint16_t	色成分積算する領域 3 の高さ
	bias_r	int8_t	画像のバイアス補正値（R 成分）（-128～127）
	bias_g	int8_t	画像のバイアス補正値（G 成分）（-128～127）
	bias_b	int8_t	画像のバイアス補正値（B 成分）（-128～127）
	gain_r	uint16_t	画像のゲイン補正値（R 成分） 上位 4bit が整数部、下位 12bit が小数部
	gain_g	uint16_t	画像のゲイン補正値（G 成分） 上位 4bit が整数部、下位 12bit が小数部
	gain_b	uint16_t	画像のゲイン補正値（B 成分） 上位 4bit が整数部、下位 12bit が小数部
	blend	uint16_t	ノイズ除去の強度（0x000～0x100） 0x000：OFF、0x100：ON（最大）
	threshold	uint8_t	二値化の閾値（0～255）
	dilate	uint8_t	Dilate 回数（0～60、2 の整数倍）
	erode	uint8_t	Erode 回数（0～60、2 の整数倍）
タイル数	1) 6 、2) 4		
分割処理	不可		

解説

入力画像

「4.2.3 Simple ISP API」の入力画像と同じです。

出力画像

dst1 領域に、二値化（8BPP）データが、パラメータ width、height で指定した画像サイズで出力されます。

dst2 領域に、ノイズ除去処理まで実施したグレイスケール（8BPP）データが、パラメータ width、height で指定した画像サイズで出力されます。

各パイプライン処理詳細

■色成分補正

「4.2.3 Simple ISP API」の色成分補正と同じです。

■色成分積算

「4.2.3 Simple ISP API」の色成分積算と同じです。

■デモザイク

線形補間法（LI）により、ベイヤー配列からグレイスケールへ変換します。線形補間法は偽色が発生する可能性があり、発生した偽色は二値化を行う際細かな点状のノイズとなる可能性があります。しかし、パラメータ dst1 に出力される二値画像は、Dilate、Erode 処理で細かな点状のノイズを除去することが可能であり、パラメータ dst2 に出力されるグレイスケール画像は、色成分を含まないため影響が小さいので、処理の軽い線形補間法を使用しています。

■ノイズ除去

「4.2.3 Simple ISP API」のノイズ除去と同じです。

■Sobel

Sobel フィルタを用いてエッジの強調を行います。
Sobel フィルタについては、4.10.4 Sobel を参照してください。

■二値化

閾値（threshold）で二値化します。入力データが閾値を超えた場合は 255、閾値以下の場合は 0 にします。

■Dilate

Dilate 処理をパラメータ dilate に指定した回数分行います。
Dilate 処理の詳細については、4.10.7 Dilate を参照してください。

■Erode

Erode 処理をパラメータ erode に指定した回数分行います。
Erode 処理の詳細については、4.10.8 Erode を参照してください。

使用例

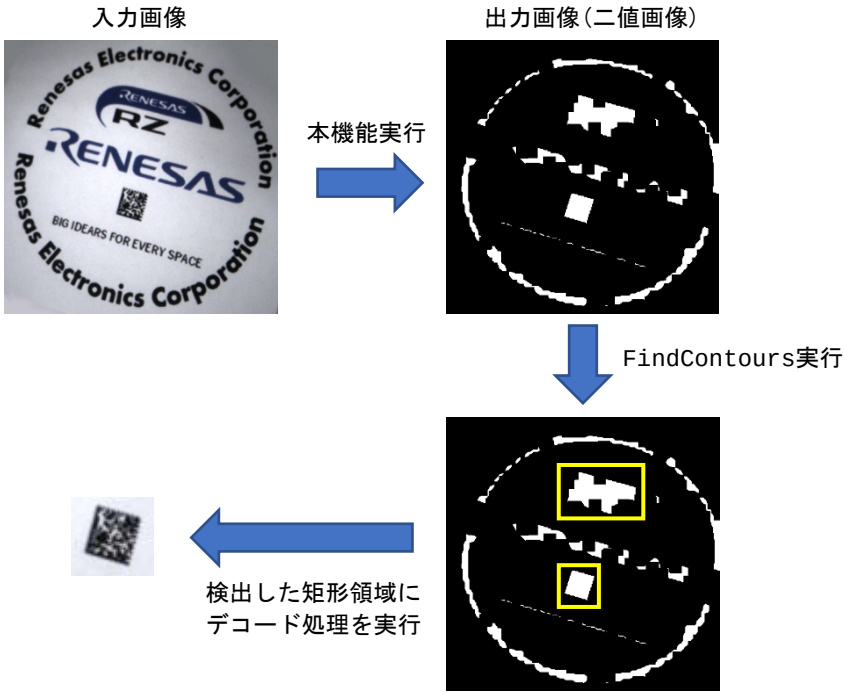
■オブジェクト検出の例

本機能では、複雑な輪郭を持つオブジェクトの輪郭を強調することで、オブジェクト検出に適した画像を作成します。本機能を 2D Barcode のデコードの前処理に使用する例を以下に示します。本機能を使用すると、全画面を検索することなく 2D Barcode の存在する可能性のある場所のみをデコードすることで、短時間で 2D Barcode の位置を検出することができます。画像サイズや環境光によって、パラメータ threshold、dilate、erode を調整する必要があります。また、検索対象の 2D Barcode の大きさを考慮して FindContours のパラメータ dst_rect_size、threshold_width、threshold_height を設定してください。

[パラメータ設定例]

```
・Simple ISP with object detection using sobel(本機能)
   width     = 480
   height    = 480
   threshold = 64
   dilate    = 4
   erode     = 10

・FindContours
   dst_rect_size    = 15
   threshold_width  = 36
   threshold_height = 36
```



■露光制御の例

「4.2.3 Simple ISP API」の露光制御の例と同じです。

■ホワイトバランスの例

「4.2.3 Simple ISP API」のホワイトバランスの例と同じです。

注意	なし

4.6　Simple ISP with distortion correction

4.6.1　概要

本機能は、監視カメラやドアフォンの近距離撮影などに使用される広角レンズの樽型歪み補正を行う Simple ISP です。本機能は、メモリ上にあるキャプチャーデータ（ベイヤー配列）に対して、色成分補正、色成分積算、デモザイク、ノイズ除去、鮮鋭化、ガンマ補正、歪み補正をパイプライン処理し、グレイスケール画像を出力します。なお、AE（自動露光制御）は、本機能から得た色成分積算値を用い、CPU 側で CMOS センサのゲインやシャッター速度を調整することで実現できます。

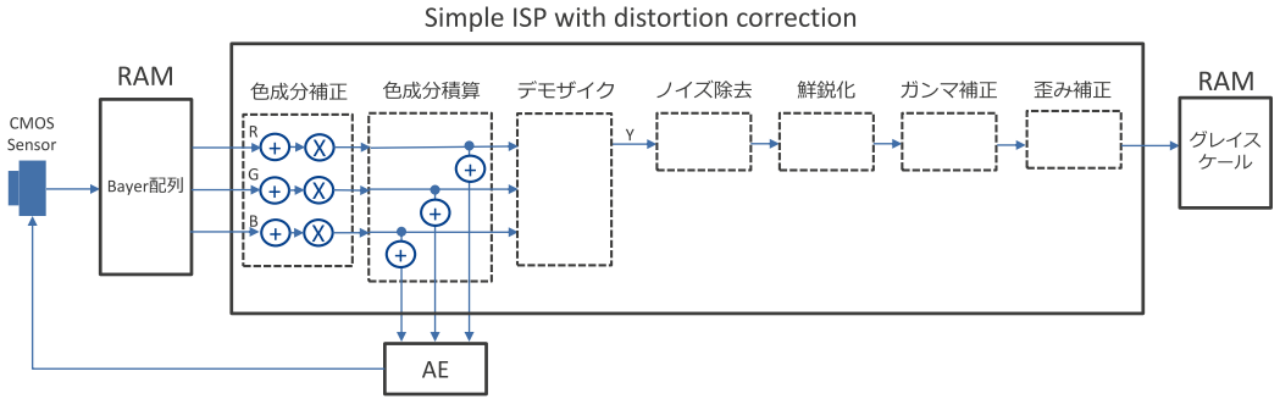


図 4.6　Simple ISP with distortion correction ブロック図

色成分補正　　：ベイヤー配列の各 RGB 成分に対して加算と乗算による補正処理
色成分積算　　：ベイヤー配列の各 RGB 成分に対しての積算値算出
デモザイク　　：ベイヤー配列から Y 成分への補間処理（LI）
ノイズ除去　　：Y 成分に対するノイズ除去処理（Median filter）
鮮鋭化　　：Y 成分に対する鮮鋭化処理（Unsharp masking）
ガンマ補正　　：Y 成分に対するガンマ補正処理
歪み補正　　：樽型歪み補正処理

4.6.2 API

Simple ISP with distortion correction

樽型歪み補正を行う Simple ISP です

コンフィグレーションデータファイル	r_drp_simple_isp_distortion_correction_6.dat
対応バージョン	1.00
コンフィグレーションデータサイズ（バイト）	699776
ヘッダファイル	r_drp_simple_isp_distortion_correction.h

パラメータ　構造体名

r_drp_simple_isp_distortion_correction_t

メンバ名	型	説明
src	uint32_t	入力画像のアドレス
dst	uint32_t	出力画像のアドレス
width	uint16_t	画像の幅（16～1280、2 の整数倍）
height	uint16_t	画像の高さ（4～960、8 の整数倍）
component	uint8_t	1：色成分積算を取得する、0：色成分積算を取得しない
accumulate	uint32_t	色成分積算値の格納先のアドレス
area1_offset_x	uint16_t	色成分積算する領域 1 の開始位置の X 座標
area1_offset_y	uint16_t	色成分積算する領域 1 の開始位置の Y 座標
area1_width	uint16_t	色成分積算する領域 1 の幅
area1_height	uint16_t	色成分積算する領域 1 の高さ
area2_offset_x	uint16_t	色成分積算する領域 2 の開始位置の X 座標
area2_offset_y	uint16_t	色成分積算する領域 2 の開始位置の Y 座標
area2_width	uint16_t	色成分積算する領域 2 の幅
area2_height	uint16_t	色成分積算する領域 2 の高さ
area3_offset_x	uint16_t	色成分積算する領域 3 の開始位置の X 座標
area3_offset_y	uint16_t	色成分積算する領域 3 の開始位置の Y 座標
area3_width	uint16_t	色成分積算する領域 3 の幅
area3_height	uint16_t	色成分積算する領域 3 の高さ
bias_r	int8_t	画像のバイアス補正値（R 成分）（-128～127）
bias_g	int8_t	画像のバイアス補正値（G 成分）（-128～127）
bias_b	int8_t	画像のバイアス補正値（B 成分）（-128～127）
gain_r	uint16_t	画像のゲイン補正値（R 成分） 上位 4bit が整数部、下位 12bit が小数部
gain_g	uint16_t	画像のゲイン補正値（G 成分） 上位 4bit が整数部、下位 12bit が小数部
gain_b	uint16_t	画像のゲイン補正値（B 成分） 上位 4bit が整数部、下位 12bit が小数部
blend	uint16_t	ノイズ除去の強度（0x000～0x100） 0x000：OFF、0x100：ON（最大）
strength	uint8_t	鮮鋭化フィルタの強調度の値（0～255）
coring	uint8_t	鮮鋭化フィルタのコアリングの値（0～255）
gamma	uint8_t	1：ガンマ補正あり、0：ガンマ補正なし
table	uint32_t	ガンマ補正に使う LUT アドレス
work	uint32_t	ワークエリアのアドレス 歪み補正処理で使用する領域です。 データサイズ：(width) × (distortion_wl) + 2 バイト（最大値）
distortion	uint16_t	樽型歪み補正係数（0～65535） 3000 以下の値を推奨します。
distortion_wl	uint16_t	ワークエリアの高さ （(height)/2+|(distortion_oy)|、4 の整数倍）

	distortion_ox	int16_t	樽型歪み補正の中心座標の X 座標値—入力画像の中心の X 座標値（width の±20%の値を推奨します） 詳細は解説を参照してください。
	distortion_oy	int16_t	樽型歪み補正の中心座標の Y 座標値—入力画像の中心の Y 座標値（height の±20%の値を推奨します） 詳細は解説を参照してください。
タイル数	6		
分割処理	不可		

解説

入力画像

「4.2.3 Simple ISP API」の入力画像と同じです。

出力画像

dst 領域に、グレイスケール（8BPP）データが、パラメータ width、height で指定した画像サイズで出力されます。

各パイプライン処理詳細

■色成分積算

「4.2.3 Simple ISP API」の色成分積算と同じです。

■色成分補正

「4.2.3 Simple ISP API」の色成分補正と同じです。

■デモザイク

線形補間法（LI）により、ベイヤー配列からグレイスケールへ変換します。

■ノイズ除去

「4.2.3 Simple ISP API」のノイズ除去と同じです。

■鮮鋭化

「4.2.3 Simple ISP API」の鮮鋭化と同じです。

■ガンマ補正

「4.2.3 Simple ISP API」のガンマ補正と同じです。

■歪み補正

本機能の歪み補正は、中心部分が膨らむように樽型に歪んだ画像の補正を行います。樽型歪みは、中心は比較的歪みが少なく、周辺にかけて歪みが大きくなるため、中心点からの距離が遠い点ほど外側に移動することで、画像の樽型歪みを補正します。本機能は、歪み補正の例に示す程度の歪みの大きさを想定しています。魚眼レンズ等の歪みの大きい画像の補正には向いていません。

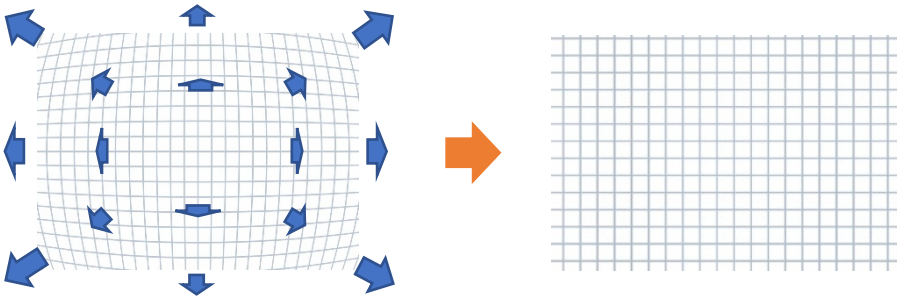
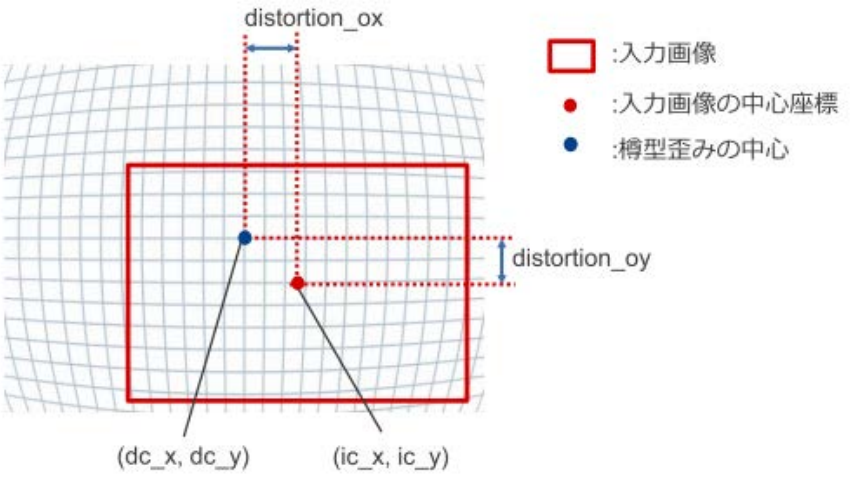


図 4.7　　歪み補正の模式図

歪み補正用のパラメータとして、distortion_ox、distortion_oy には、入力画像の中心座標と樽型歪みの中心の差を指定します。入力画像の中心座標を(ic_x,ic_y)、樽型歪みの中心座標を(dc_x,dc_y)としたとき、distortion_ox と distortion_oy の値は以下のようになります。

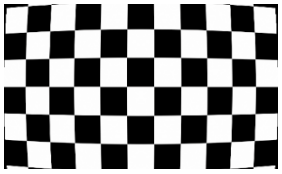
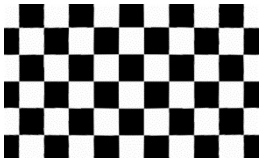
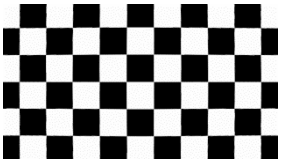
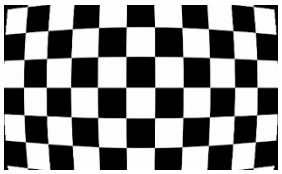
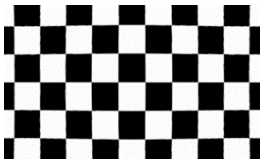
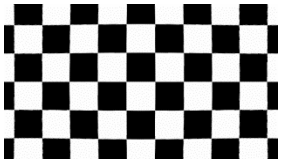
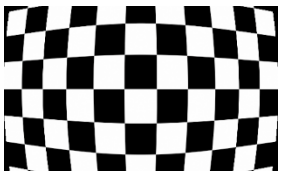
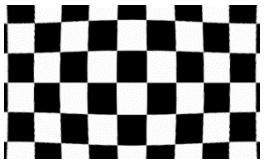
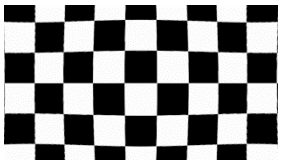
```
distortion_ox = dc_x – ic_x
distortion_oy = dx_y – ic_y
```



使用例

■歪み補正の例

本機能の歪み補正は、パラメータ distortion、distortion_ox、distortion_oy の値を調整することで画像の歪みを補正します。チェスボード等の歪みが分かりやすい画像を撮影し、歪みのない出力画像となるまでパラメータの値を調整してください。
参考までに、本機能の歪み補正を実施した例を以下に示します。

歪み	入力画像	歪み補正後の出力画像	
		画像サイズ(800*480)	画像サイズ(1280*720)
小 ▲		 distortion=1600	 distortion=600
		 distortion=2300	 distortion=950
▼ 大		 distortion=3000	 distortion=1350

上記は、distortion_ox、distortion_oy が 0 の場合の例です。上記の distortion の値を参考に設定しても歪みが補正できない場合、樽型歪みの中心がずれている可能性があります。
distortion_ox、distortion_oy の値を変更して補正を行ってください。

■露光制御の例

「4.2.3 Simple ISP API」の露光制御の例と同じです。

■ホワイトバランスの例

「4.2.3 Simple ISP API」のホワイトバランスの例と同じです。

注意　なし

4.7　Simple ISP with scaling and normalization (32bit)

4.7.1　概要

本機能は AI の推論の前処理用の Simple ISP です。前処理というのは、浮動小数点化と正規化とリサイズを指しています。本機能を用いることによってユーザーは CMOS センサの入力から AI の推論の前処理を行った画像を得るまでを本機能一つで実現できます。本機能は、メモリ上にあるキャプチャーデータ（ベイヤー配列）に対して、色成分補正、色成分積算、デモザイク、ノイズ除去、鮮鋭化、ガンマ補正、カラー変換、リサイズ、正規化、浮動小数点化等をパイプライン処理し、表示用 ARGB 画像と AI 用 RGB 画像を出力します。なお、AE（自動露光制御）は、本機能から得た色成分積算値を用い、CPU 側で CMOS センサのゲインやシャッター速度を調整することで実現できます。

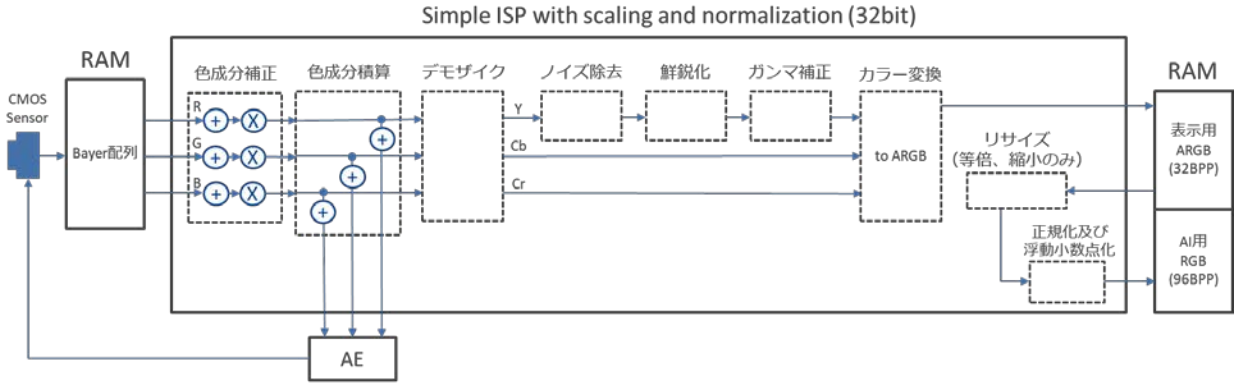


図 4.8　Simple ISP with scaling and normalization (32bit)ブロック図

色成分補正	: ベイヤー配列の各 RGB 成分に対して加算と乗算による補正処理
色成分積算	: ベイヤー配列の各 RGB 成分に対しての積算値算出
デモザイク	: ベイヤー配列から YCbCr 成分への補間処理（ACPI）
ノイズ除去	: Y 成分に対するノイズ除去処理（Median filter）
鮮鋭化	: Y 成分に対する鮮鋭化処理（Unsharp masking）
ガンマ補正	: Y 成分に対するガンマ補正処理
カラー変換	: YCbCr 成分に対する ARGB 変換処理（A 部分は 255 固定）
リサイズ	: RGB 成分に対するリサイズ処理（等倍、縮小のみ）（Bilinear interpolation）
正規化・浮動小数点化	: リサイズ結果に対する正規化及び浮動小数点化処理

4.7.2 API

Simple ISP with scaling and normalization (32bit)

AI 推論の前処理（浮動小数点化、正規化、リサイズ）を行う Simple ISP です

コンフィグレーションデータファイル	r_drp_simple_isp_scal_normaliz_b32_6.dat
対応バージョン	1.00
コンフィグレーションデータサイズ（バイト）	691840
ヘッダファイル	r_drp_simple_isp_scal_normaliz_b32.h

パラメータ　構造体名

r_drp_simple_isp_scal_normaliz_b32_t

メンバ名	型	説明
src	uint32_t	入力画像のアドレス
dst_mode	uint8_t	0: 出力画像（AI 用 RGB 画像）を Packed format で出力する 1: 出力画像（AI 用 RGB 画像）を Planar format で出力する
dst1	uint32_t	出力画像（表示用 ARGB 画像）のアドレス
dst2	uint32_t	出力画像（AI 用 RGB 画像）のアドレス dst_mode=0 の場合 Packed format で RGB 成分を出力する dst_mode=1 の場合 Planar format の R 成分を出力する
dst3	uint32_t	出力画像（AI 用 RGB 画像）のアドレス dst_mode=0 の場合出力しない dst_mode=1 の場合 Planar format の G 成分を出力する
dst4	uint32_t	出力画像（AI 用 RGB 画像）のアドレス dst_mode=0 の場合出力しない dst_mode=1 の場合 Planar format の B 成分を出力する
src_width	uint16_t	入力画像の幅（16～1920、2 の整数倍）
src_height	uint16_t	入力画像の高さ（4～1080、2 の整数倍）
dst_width	uint16_t	リサイズ後の画像の横幅（16～src_width、2 の整数倍）
dst_height	uint16_t	リサイズ後の画像の縦幅（4～src_height、2 の整数倍）
component	uint8_t	1：色成分積算を取得する、0：色成分積算を取得しない
accumulate	uint32_t	色成分積算値の格納先のアドレス
area1_offset_x	uint16_t	色成分積算する領域 1 の開始位置の X 座標
area1_offset_y	uint16_t	色成分積算する領域 1 の開始位置の Y 座標
area1_width	uint16_t	色成分積算する領域 1 の幅
area1_height	uint16_t	色成分積算する領域 1 の高さ
area2_offset_x	uint16_t	色成分積算する領域 2 の開始位置の X 座標
area2_offset_y	uint16_t	色成分積算する領域 2 の開始位置の Y 座標
area2_width	uint16_t	色成分積算する領域 2 の幅
area2_height	uint16_t	色成分積算する領域 2 の高さ
area3_offset_x	uint16_t	色成分積算する領域 3 の開始位置の X 座標
area3_offset_y	uint16_t	色成分積算する領域 3 の開始位置の Y 座標
area3_width	uint16_t	色成分積算する領域 3 の幅
area3_height	uint16_t	色成分積算する領域 3 の高さ
bias_r	int8_t	画像のバイアス補正値（R 成分）（-128～127）
bias_g	int8_t	画像のバイアス補正値（G 成分）（-128～127）
bias_b	int8_t	画像のバイアス補正値（B 成分）（-128～127）
gain_r	uint16_t	画像のゲイン補正値（R 成分） 上位 4bit が整数部、下位 12bit が小数部
gain_g	uint16_t	画像のゲイン補正値（G 成分） 上位 4bit が整数部、下位 12bit が小数部

	gain_b	uint16_t	画像のゲイン補正値（B 成分） 上位 4bit が整数部、下位 12bit が小数部
	blend	uint16_t	ノイズ除去の強度（0x000～0x100） 0x000：OFF、0x100：ON（最大）
	strength	uint8_t	鮮鋭化フィルタの強調度の値（0～255）
	coring	uint8_t	鮮鋭化フィルタのコアリングの値（0～255）
	gamma	uint8_t	1：ガンマ補正あり、0：ガンマ補正なし
	table	uint32_t	ガンマ補正に使う LUT アドレス
	r_table	uint32_t	R 成分の正規化及び浮動小数点化に使う LUT アドレス
	g_table	uint32_t	G 成分の正規化及び浮動小数点化に使う LUT アドレス
	b_table	uint32_t	B 成分の正規化及び浮動小数点化に使う LUT アドレス
タイル数	6		
分割処理	不可		

解説

入力画像

「4.2.3 Simple ISP API」の入力画像と同じです。

出力画像

dst1 領域に、ARGB（32BPP）データが、パラメータ src_width、src_height で指定した画像サイズで出力されます。

ARGB の出力フォーマット：
A0,R0,G0,B0,A1,R1,G1,B1・・・・・
（An, Rn, Gn, Bn: n 番目のピクセルの各色の明度、An は 255 固定）

dst_mode=0 の場合は、dst2 領域に、色成分毎の LUT によって正規化及び浮動小数点化された RGB（96BPP）データが、パラメータ dst_width、dst_height で指定した画像サイズで出力されます。

dst_mode=0 の場合の dst2 の RGB 出力フォーマット：
R0,G0,B0,R1,G1,B1・・・・・
（Rn, Gn, Bn: n 番目のピクセルの各色の明度）

dst_mode=1 の場合は、色成分毎の LUT によって正規化及び浮動小数点化された RGB（96BPP）データが、パラメータ dst_width、dst_height で指定した画像サイズで、dst2 領域に R 成分、dst3 領域に G 成分、dst4 領域に B 成分が出力されます。

dst_mode=1 の場合の dst2 の出力フォーマット：
R0,R1,R2,R3・・・・・
dst_mode=1 の場合の dst3 の出力フォーマット：
G0,G1,G2,G3・・・・・
dst_mode=1 の場合の dst4 の出力フォーマット：
B0,B1,B2,B3・・・・・
（Rn, Gn, Bn: n 番目のピクセルの各色の明度）

各パイプライン処理詳細

■色成分積算

「4.2.3 Simple ISP API」の色成分積算と同じです。

■色成分補正

「4.2.3 Simple ISP API」の色成分補正と同じです。

■デモザイク

適応型カラープレーン補間法（ACPI）により、ベイヤー配列から YCbCr422 へ変換します。

■ノイズ除去

「4.2.3 Simple ISP API」のノイズ除去と同じです。

■鮮鋭化

「4.2.3 Simple ISP API」の鮮鋭化と同じです。

■ガンマ補正

「4.2.3 Simple ISP API」のガンマ補正と同じです。

■カラー変換

YCbCr を ARGB へ変換します。

■リサイズ

dst1 領域に出力された画像の RGB 成分に対し、bilinear interpolation アルゴリズムを用いて、dst_width、dst_height で指定したサイズにリサイズを行います。
詳細については、4.9.13 ResizeBilinear を参照してください。

■正規化・浮動小数点化

リサイズ後の各画素の RGB 値「0～255」に対し、変換テーブルのデータ参照によって正規化及び浮動小数点化を行います。

パラメータ r_table で指定したアドレスに R 成分の変換テーブル、パラメータ g_table で指定したアドレスに G 成分の変換テーブル、パラメータ b_table で指定したアドレスに B 成分の変換テーブルを格納してください。0～255 の table で指定された LUT を参照して画素値の変換を行います。

使用例

■正規化・浮動小数点化用変換テーブル作成例

正規化及び浮動小数点化する変換テーブルの作成方法例を以下に示します。
以下は、0～255 のピクセルデータを 0.0～1.0 の範囲に変換し、平均 0.485、標準偏差 0.229 のデータを入力したときに、平均 0、分散 1 の分布に変換する例です。

```
int luminance;
float mean = 0.485;
float stddev = 0.229;
float normalize;

for(luminance = 0;luminance < 256;luminance++)
{
    normalize = ((float)luminance / 255 - mean) / stddev; // min-max and z-score normalization
}
```

■露光制御の例

「4.2.3 Simple ISP API」の露光制御の例と同じです。

■ホワイトバランスの例

「4.2.3 Simple ISP API」のホワイトバランスの例と同じです。

注意　なし

4.8　Simple ISP with color calibration and 3DNR

4.8.1　概要

本機能は、カラーマトリクス補正や 3D ノイズリダクションにより、色再現性の高い画像を出力することに特化した Simple ISP です。これを用いることによってユーザーは、AI 処理に適した色再現性の高い画像や、人が見てより自然な色味と感じる画像を得ることができます。本機能は、メモリ上にあるキャプチャーデータ（ベイヤー配列）に対して、バイアス補正、色成分積算、デモザイク、ゲイン補正、カラーマトリクス補正、ノイズ除去、鮮鋭化、ガンマ補正、3D ノイズリダクションをパイプライン処理し、YCbCr422 形式の画像を出力します。なお、AE（自動露光制御）は、本機能から得た色成分積算値を用い、CPU 側で CMOS センサのゲインやシャッター速度を調整することで実現できます。

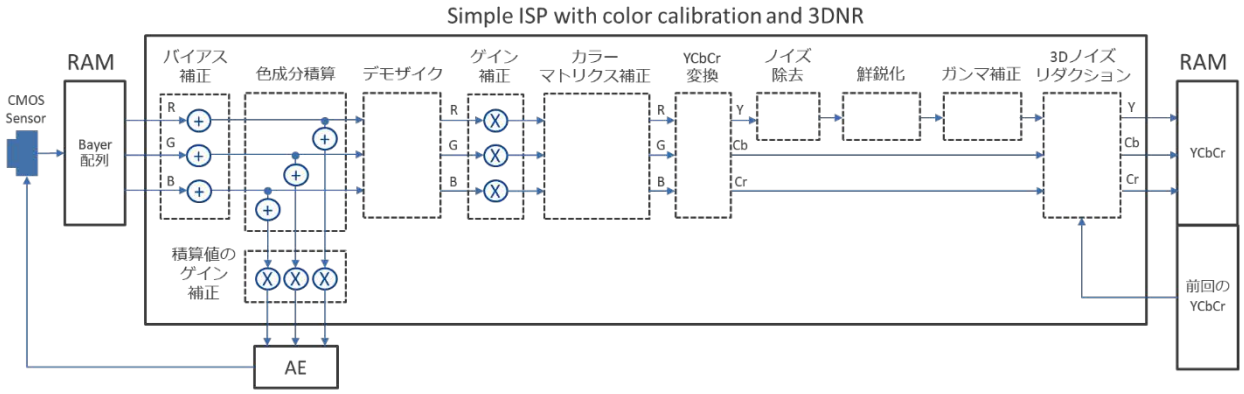


図 4.9　Simple ISP with color calibration and 3DNR ブロック図

バイアス補正	: ベイヤー配列の各 RGB 成分に対して加算による補正処理
色成分積算	: ベイヤー配列の各 RGB 成分に対しての積算値を算出
積算値のゲイン補正	: ベイヤー配列の各 RGB 成分の積算値に対して乗算による補正処理
デモザイク	: ベイヤー配列から RGB 成分への補完処理
ゲイン補正	: デモザイク後の RGB 成分に対して乗算による補正処理
カラーマトリクス補正	: RGB 成分に対する 3×3 変換行列による色補正処理
YCbCr 変換	: RGB 成分に対する YCbCr 成分への変換処理
ノイズ除去	: Y 成分に対するノイズ除去処理（Median filter）
鮮鋭化	: Y 成分に対する鮮鋭化処理（Unsharp masking）
ガンマ補正	: Y 成分に対するガンマ補正処理
3D ノイズリダクション	: 前回の YCbCr 画像を利用した YCbCr 成分に対するノイズリダクション処理

4.8.2 API

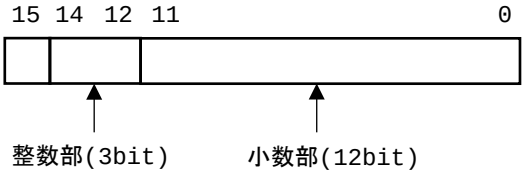
Simple ISP with color calibration and 3DNR

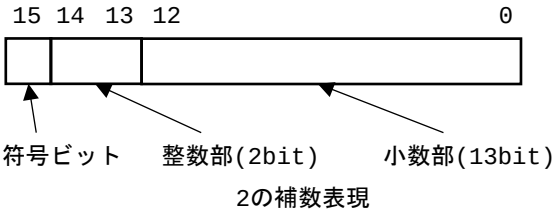
カラーマトリクス補正や 3D ノイズリダクションにより、色再現性の高い画像を出力することに特化した Simple ISP です

コンフィグレーションデータファイル	r_drp_simple_isp_colcal_3dnr_6.dat
対応バージョン	1.00
コンフィグレーションデータサイズ（バイト）	560192
ヘッダファイル	r_drp_simple_isp_colcal_3dnr.h

パラメータ　構造体名

r_drp_simple_isp_colcal_3dnr_t

メンバ名	型	説明
src	uint32_t	入力画像のアドレス
prev	uint32_t	前フレーム出力画像のアドレス （0 を指定した場合は、3D ノイズリダクション補正処理は行いません）
dst	uint32_t	出力画像のアドレス
width	uint16_t	画像の幅（16～1920、2 の整数倍）
height	uint16_t	画像の高さ（4～1080、2 の整数倍）
component	uint8_t	1：色成分積算を取得する、0：色成分積算を取得しない
accumulate	uint32_t	色成分積算値の格納先のアドレス
area1_offset_x	uint16_t	色成分積算する領域 1 の開始位置の X 座標
area1_offset_y	uint16_t	色成分積算する領域 1 の開始位置の Y 座標
area1_width	uint16_t	色成分積算する領域 1 の幅
area1_height	uint16_t	色成分積算する領域 1 の高さ
area2_offset_x	uint16_t	色成分積算する領域 2 の開始位置の X 座標
area2_offset_y	uint16_t	色成分積算する領域 2 の開始位置の Y 座標
area2_width	uint16_t	色成分積算する領域 2 の幅
area2_height	uint16_t	色成分積算する領域 2 の高さ
area3_offset_x	uint16_t	色成分積算する領域 3 の開始位置の X 座標
area3_offset_y	uint16_t	色成分積算する領域 3 の開始位置の Y 座標
area3_width	uint16_t	色成分積算する領域 3 の幅
area3_height	uint16_t	色成分積算する領域 3 の高さ
bias_r	int8_t	画像のバイアス補正値（R 成分）（-128～127） -16 を推奨値としています。
bias_g	int8_t	画像のバイアス補正値（G 成分）（-128～127） -16 を推奨値としています。
bias_b	int8_t	画像のバイアス補正値（B 成分）（-128～127） -16 を推奨値としています。
gain_r	uint16_t	画像のゲイン補正値（R 成分）を固定小数化した値 固定小数のフォーマットは下図の通りです。  0x2315 を推奨値としています。

gain_g	uint16_t	画像のゲイン補正値（G 成分）を固定小数化した値 固定小数のフォーマットは gain_r と同じです。 0x1800 を推奨値としています。
gain_b	uint16_t	画像のゲイン補正値（B 成分）を固定小数化した値 固定小数のフォーマットは gain_r と同じです。 0x2a64 を推奨値としています。
blend	uint16_t	ノイズ除去の強度（0x000～0x100） 0x000：OFF、0x100：ON（最大）
strength	uint8_t	鮮鋭化フィルタの強調度の値（0～255）
coring	uint8_t	鮮鋭化フィルタのコアリングの値（0～255）
gamma	uint8_t	1：ガンマ補正あり、0：ガンマ補正なし
table	uint32_t	ガンマ補正に使う LUT アドレス
matrix_c11	int16_t	カラーマトリクス補正用変換行列における 1 行 1 列目の要素を固定小数化した値 固定小数のフォーマットは下図の通りです。  2の補数表現 0x3b8b を推奨値としています。
matrix_c12	int16_t	カラーマトリクス補正用変換行列における 1 行 2 列目の要素を固定小数化した値 固定小数のフォーマットは matrix_c11 と同じです。 0xeaf1 を推奨値としています。
matrix_c13	int16_t	カラーマトリクス補正用変換行列における 1 行 3 列目の要素を固定小数化した値 固定小数のフォーマットは matrix_c11 と同じです。 0xf405 を推奨値としています。
matrix_c21	int16_t	カラーマトリクス補正用変換行列における 2 行 1 列目の要素を固定小数化した値 固定小数のフォーマットは matrix_c11 と同じです。 0xf726 を推奨値としています。
matrix_c22	int16_t	カラーマトリクス補正用変換行列における 2 行 2 列目の要素を固定小数化した値 固定小数のフォーマットは matrix_c11 と同じです。 0x2fdf を推奨値としています。
matrix_c23	int16_t	カラーマトリクス補正用変換行列における 2 行 3 列目の要素を固定小数化した値 固定小数のフォーマットは matrix_c11 と同じです。 0xf84d を推奨値としています。
matrix_c31	int16_t	カラーマトリクス補正用変換行列における 3 行 1 列目の要素を固定小数化した値 固定小数のフォーマットは matrix_c11 と同じです。 0x0182 を推奨値としています。
matrix_c32	int16_t	カラーマトリクス補正用変換行列における 3 行 2 列目の要素を固定小数化した値 固定小数のフォーマットは matrix_c11 と同じです。 0xc9c5 を推奨値としています。

	matrix_c33	int16_t	カラーマトリクス補正用変換行列における 3 行 3 列目の要素を固定小数化した値 固定小数のフォーマットは matrix_c11 と同じです。 0x5c3a を推奨値としています。
	y_coef	uint8_t	3D ノイズリダクションにおける Y 動き情報を計算する際の Y 信号差分の比率を示す係数（0～64） 64 を推奨値としています。 詳細は解説を参照してください。
	c_coef	uint8_t	3D ノイズリダクションにおける C 動き情報を計算する際の Y 信号差分の比率を示す係数（0～64） 32 を推奨値としています。 詳細は解説を参照してください。
	y_alpha_max	uint8_t	3D ノイズリダクションにおける Y 相関係数の最大値（0～255） 128 を推奨値としています。 詳細は解説を参照してください。
	y_thresh_a	uint16_t	3D ノイズリダクションにおいて Y 相関係数が最大値となる Y 動き情報の閾値（0～511） 8 を推奨値としています。 詳細は解説を参照してください。
	y_thresh_b	uint16_t	3D ノイズリダクションにおいて Y 相関係数が 0 となる Y 動き情報の閾値（0～511） 16 を推奨値としています。 詳細は解説を参照してください。
	y_tilt	uint16_t	3D ノイズリダクションにおいて Y 相関係数算出に使用する比例定数（0～4095） 512 を推奨値としています。 詳細は解説を参照してください。
	c_alpha_max	uint8_t	3D ノイズリダクションにおいて C 相関係数の最大値（0～255） 128 を推奨値としています。 詳細は解説を参照してください。
	c_thresh_a	uint16_t	3D ノイズリダクションにおいて C 相関係数が最大値となる C 動き情報の閾値（0～511） 8 を推奨値としています。 詳細は解説を参照してください。
	c_thresh_b	uint16_t	3D ノイズリダクションにおける C 相関係数が 0 となる C 動き情報の閾値（0～511） 16 を推奨値としています。 詳細は解説を参照してください。
	c_tilt	uint16_t	3D ノイズリダクションにおいて C 相関係数算出に使用する比例定数（0～4095） 512 を推奨値としています。 詳細は解説を参照してください。
タイル数	6		
分割処理	不可		

解説

入力画像

src 領域については、「4.2.3 Simple ISP API」の入力画像と同じです。

出力画像

dst 領域に YCbCr422（16BPP）データがパラメータ width、height で指定した画像サイズで出力されます。

YCbCr422 の出力フォーマット：
CB0,Y0,CR0,Y1,CB2,Y2,CR2,Y3・・・・・
（Yn, CBn, CRn: n 番目のピクセルの輝度と色差の値）

前フレーム出力画像

3D ノイズリダクションを使用する場合は、prev 領域に dst 領域と同じ画像サイズのアドレスを指定します。
prev と dst は同一のアドレスを指定することが可能です。prev と dst に同一のアドレスを指定する場合は、メモリを節約することができますが、dst 領域のデータを書き換えることができません。
prev と dst を別のアドレスで指定する場合、dst 領域のデータは書き換え可能ですが、prev 領域のデータを書き換えることができません。

3D ノイズリダクション処理では、入力画像（src）と前フレーム出力画像（prev）を使用し、動き情報と相関係数を計算して混合処理を行うことで、高感度ノイズを低減しています。

各パイプライン処理詳細

■バイアス補正

ベイヤー配列の RGB の各成分に対し、パラメータ bias_r、bias_g、bias_b で設定した値が加算されます。

■色成分積算

「4.2.3 Simple ISP API」の色成分積算と同じです。

■デモザイク

適応型カラープレーン補間法（ACPI）により、ベイヤー配列から RGB へ変換します。

■ゲイン補正

デモザイクにより変換した RGB の各成分や、ベイヤー配列の RGB 各成分に対し積算した結果に対し、パラメータ gain_r、gain_g、gain_b で設定した値が乗算されます。

■カラーマトリクス補正
RGB の各成分 R_{in}、G_{in}、B_{in} に対し、3×3 変換行列を下図のように用いて、R_{out}、G_{out}、B_{out} に変換します。

$$\begin{pmatrix} R_{out} \\ G_{out} \\ B_{out} \end{pmatrix} = \begin{pmatrix} matrix\_c11 & matrix\_c12 & matrix\_c13 \\ matrix\_c21 & matrix\_c22 & matrix\_c23 \\ matrix\_c31 & matrix\_c32 & matrix\_c33 \end{pmatrix} \begin{pmatrix} R_{in} \\ G_{in} \\ B_{in} \end{pmatrix}$$

■YCbCr 変換

RGB を YCbCr へ変換します。

■ノイズ除去

「4.2.3 Simple ISP API」のノイズ除去と同じです。

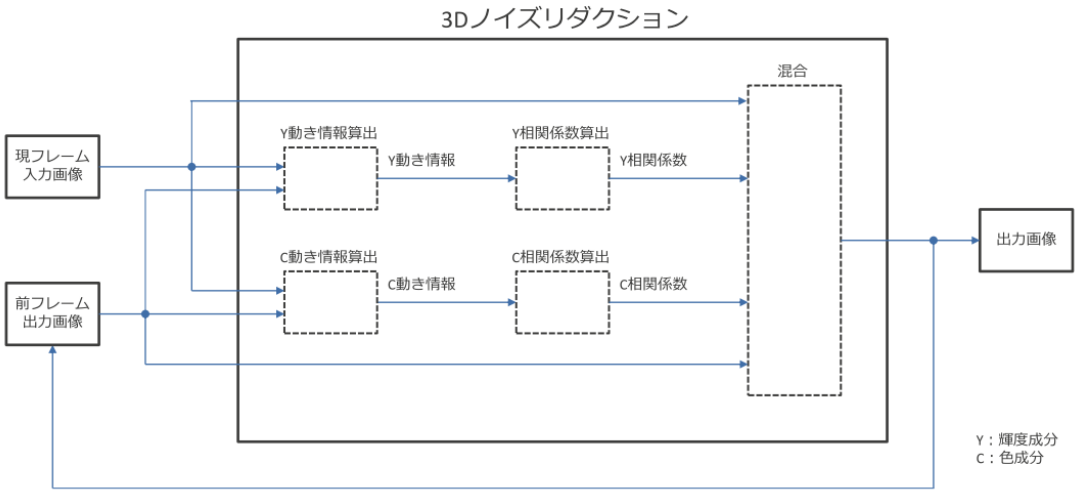
■鮮鋭化

「4.2.3 Simple ISP API」の鮮鋭化と同じです。

■ガンマ補正

「4.2.3 Simple ISP API」のガンマ補正と同じです。

■3D ノイズリダクション
本機能は、動画における高感度ノイズの低減を目的としています。現フレーム入力画像と前フレーム出力画像に対し、動き情報と相関係数を計算して混合処理を行います。以下に、処理内容のブロック図と詳細を記載します。



①Y 動き情報算出
n 番目のピクセルの輝度 Yn における Y 動き情報を、以下の計算式に従い算出します。

```
Y 動き情報 = (Y 信号差分 × y_coef + C 信号差分 × (64 - y_coef)) ÷ 64
```

・Y 信号差分 = |現フレーム入力 Yn - 前フレーム出力 Yn|

・C 信号差分 = |現フレーム入力 CB((n>>1)<<1) - 前フレーム出力 CB((n>>1)<<1)| +
|現フレーム入力 CR((n>>1)<<1) - 前フレーム出力 CR((n>>1)<<1)|

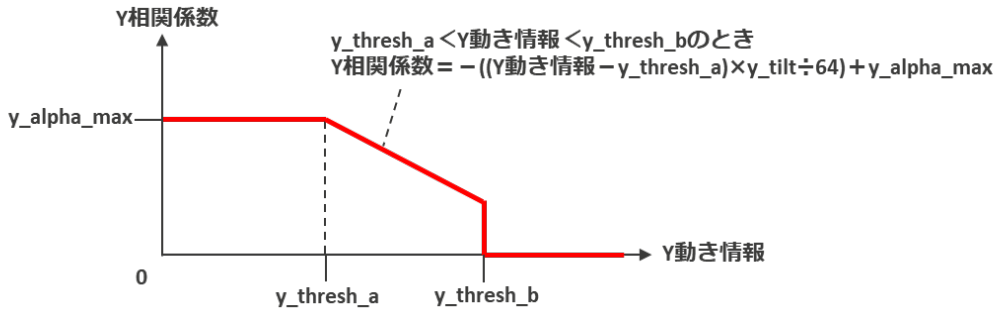
②C 動き情報算出
n 番目のピクセルの色差 CBn,CRn における C 動き情報を、以下の計算式に従い算出します。

```
C 動き情報 = (Y 信号差分 × c_coef + C 信号差分 × (64 - c_coef)) ÷ 64
```

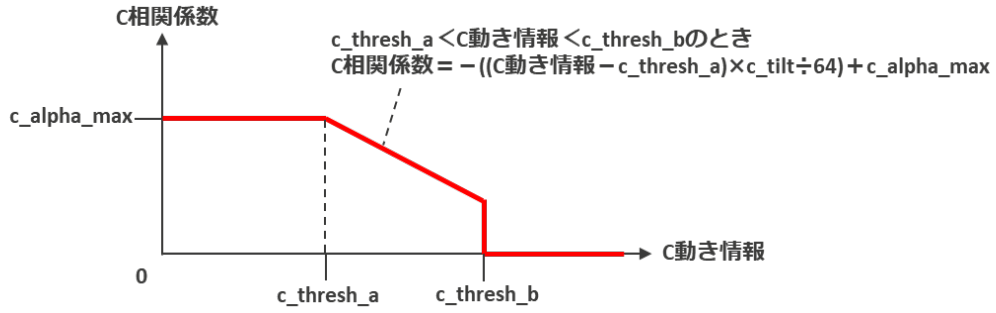
・Y 信号差分 = (|現フレーム入力 Yn - 前フレーム出力 Yn| +
|現フレーム入力 Y(n+1) - 前フレーム出力 Y(n+1)|) ÷ 2

・C 信号差分 = |現フレーム入力 CBn - 前フレーム出力 CBn| +
|現フレーム入力 CRn - 前フレーム出力 CRn|

③Y 相関係数算出
①で計算した Y 動き情報から、現フレーム入力画像と前フレーム出力画像の Yn を混合する割合を示す Y 相関係数を、下のグラフに基づき算出します。



④C 相関係数算出
②で計算した C 動き情報から、現フレーム入力画像と前フレーム出力画像の CBn,CRn を混合する割合を示す C 相関係数を、下のグラフに基づき算出します。



⑤混合
相関係数を基に、現フレーム入力データと前フレーム出力データを以下の算出式により混合します。

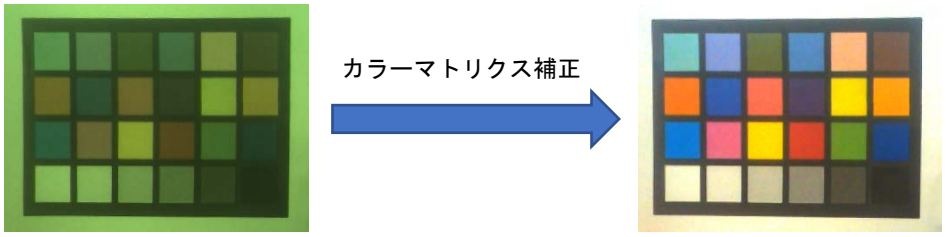
```
Y 出力データ ＝（前フレーム出力 Yn × Y 相関係数 ＋
                現フレーム入力 Yn × （256 － Y 相関係数）） ÷ 256
CB 出力データ ＝（前フレーム出力 CBn × C 相関係数 ＋
                現フレーム入力 CBn × （256 － C 相関係数）） ÷ 256
CR 出力データ ＝（前フレーム出力 CRn × C 相関係数 ＋
                現フレーム入力 CRn × （256 － C 相関係数）） ÷ 256
```

使用例

■カラーマトリクス補正の例

本機能では、3×3 変換行列を使用してカラーマトリクス補正を実施します。Raspberry Pi Camera V2 を使用し、パラメータ matrix_cmn(mn:11,12,13,21,22,23,31,32,33)に、推奨値を設定した場合の例を以下に示します。パラメータについては、使用するカメラに応じて調整してください。

```
[パラメータ設定値]
  matrix_c11 = 0x3b8b
  matrix_c12 = 0xeaf1
  matrix_c13 = 0xf405
  matrix_c21 = 0xf726
  matrix_c22 = 0x2fdf
  matrix_c23 = 0xf84d
  matrix_c31 = 0x0182
  matrix_c32 = 0xc9c5
  matrix_c33 = 0x5c3a
```

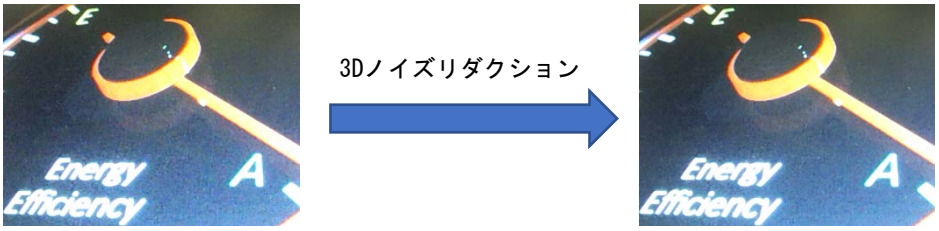


入力画像　出力画像

■3D ノイズリダクションの例

本機能の 3D ノイズリダクションでは、前回の出力画像を使用してノイズリダクションを実施します。3D ノイズリダクションを実施した例を以下に示します。本機能の 3D ノイズリダクション用パラメータの推奨値は、動画における高感度ノイズを低減するのに適した値となっているため、静止画での補正幅は小さくなります。

```
[パラメータ設定値]（推奨値を設定）
  y_coef = 64
  c_coef = 32
  y_alpha_max = 128
  y_thresh_a = 8
  y_thresh_b = 16
  y_tilt = 512
  c_alpha_max = 128
  c_thresh_a = 8
  c_thresh_b = 16
  c_tilt = 512
```



入力画像　出力画像

■露光制御の例

「4.2.3 Simple ISP API」の露光制御の例と同じです。

■ホワイトバランスの例

「4.2.3 Simple ISP API」のホワイトバランスの例と同じです。

注意　以下のパラメータの推奨値については、Raspberry Pi Camera V2 カメラを使用したときの推奨値となります。

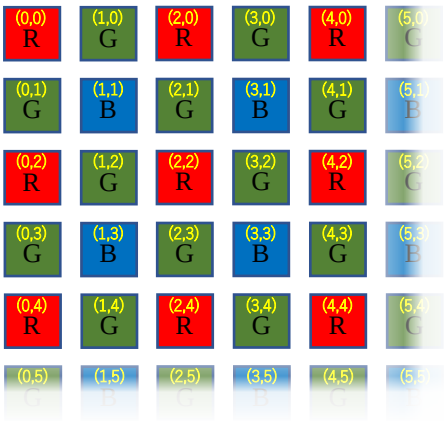
```
bias_r
bias_g
bias_b
gain_r
gain_g
gain_b
matrix_c11
matrix_c12
matrix_c13
matrix_c21
matrix_c22
matrix_c23
matrix_c31
matrix_c32
matrix_c33
```

4.9 Image transformation

4.9.1 Bayer2Grayscale

Bayer2Grayscale

CMOS カメラからの RAW データをグレイスケールへ変換します

コンフィグレーションデータファイル			r_drp_bayer2grayscale.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			59808
ヘッダファイル			r_drp_bayer2grayscale.h
パラメータ	構造体名		
	r_drp_bayer2grayscale_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	top	uint8_t	1:上端の境界処理あり。 0:上端の境界処理なし。 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
	bottom	uint8_t	1:下端の境界処理あり。 0:下端の境界処理なし。 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
入出力詳細	入力画像	アドレス　：src で指定 幅（ピクセル）　：width で指定（16～1280） 高さ（ピクセル）：height で指定（4～960） データサイズ　：(width) × (height) × 1 バイト ＜フォーマット＞ 入力画像のフォーマットは、入力画像の左上の座標を(0,0)としたときに、X,Y 座標どちらも偶数の場合は「赤」、どちらも奇数の場合は「青」、それ以外が「緑」である、下図のようなベイヤーの配列です。  上記以外のベイヤー配列は、カメラの設定を変更するか、RZ/A2M の VIN の機能を使用して対応することが可能です。詳細は解説を参照してください。	

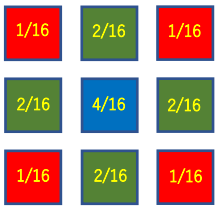
	出力画像	アドレス	: dst で指定
		幅（ピクセル）	: 入力画像と同じ
		高さ（ピクセル）	: 入力画像と同じ
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
タイル数	1		
分割処理	可		

解説

本機能は、src で指定したアドレスの画像を、ベイヤーフォーマットから 8bit のグレイスケールフォーマットに変換し、dst で指定したアドレスに出力します。

本機能は始めに入力画像を 3 × 3 フィルタを使った線形補間法で RGB に変換し、その後、RGB から Y への変換処理を行い、輝度値を算出します。

3 × 3 フィルタを使った線形補間法とは、変換したいピクセルに着目し、そのピクセルを中心とした 3 × 3 の範囲に対して、
中心のピクセルの値：4/16 倍
上下左右のピクセルの値：2/16 倍
斜め四方のピクセルの値：1/16 倍
の、それぞれの倍率を乗じて色成分ごとに合計して、ベイヤーの色密度の逆数（赤と青は 4、緑は 2）を掛ける方法で、着目ピクセルの RGB の値を求める方法です。



・中心：4/16倍
・上下左右：2/16倍
・斜め四方：1/16倍
それぞれ上記を乗じた値を色成分ごとに合計して、
色密度の逆数（赤と青は4、緑は2）を掛ける

RGB から Y への変換は、以下の式で行います。
```
Y = (Red * 76 + Green * 152 + Blue * 28) / 256
```

画面左右端のピクセルを変換する場合には、3 × 3 フィルタの一部が入力画像の領域外となり参照できないため、その代わりに 1 ライン内側のピクセルの値を参照する、複製境界（OpenCV の BORDER_REFLECT_101 境界）処理を行います。
参考 URL： https://opencv.org/

top および bottom に 1 を設定した場合は、画面の上下端も同様の複製境界処理を行います。入力画像分割を行わない場合は、top と bottom は 1 を設定して下さい。

「入出力詳細」の「入力画像」に示した図以外のベイヤー配列のカメラを使用する場合は、左上が赤色になる位置に画像を切り取ってキャプチャして下さい。画像を切り取る方法は、カメラ側の出力画像範囲をクリップする設定か、もしくは、MIPI カメラを使用する場合であれば、RZ/A2M で入力画像範囲をクリップする方法があります。後者の設定については、RZ/A2M グループ ユーザーズマニュアル ハードウェア編の 48 ビデオインプットモジュール（VIN）を参照頂くか、または、MIPI ドライバのユーザーズマニュアルで、範囲クリップ（前段）の機能を参照してください。

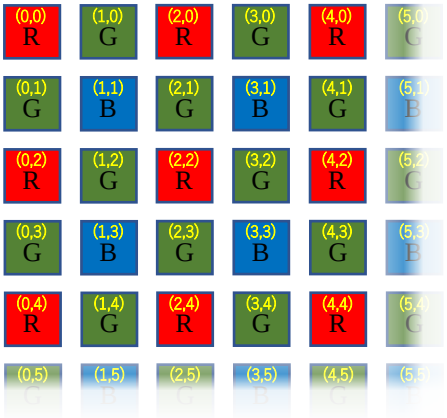
本機能は、分割処理を行わない場合、src と dst に同一アドレスを指定することが可能です。

注意　なし

4.9.2 Bayer2Rgb

Bayer2Rgb

CMOS カメラからの RAW データを RGB カラーへ変換します

コンフィグレーションデータファイル			r_drp_bayer2rgb.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			91744
ヘッダファイル			r_drp_bayer2rgb.h
パラメータ	構造体名		
	r_drp_bayer2rgb_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	top	uint8_t	1:上端の境界処理あり。 0:上端の境界処理なし。 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
	bottom	uint8_t	1:下端の境界処理あり。 0:下端の境界処理なし。 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
入出力詳細	入力画像	アドレス　：src で指定 （dst と異なるアドレスにしてください） 幅（ピクセル）　：width で指定（16～1280、2 の整数倍） 高さ（ピクセル）：height で指定（4～960、2 の整数倍） データサイズ　：(width) × (height) × 1 バイト ＜フォーマット＞ 入力画像のフォーマットは、入力画像の左上の座標を(0,0)としたときに、X,Y 座標どちらも偶数の場合は「赤」、どちらも奇数の場合は「青」、それ以外が「緑」である、下図のようなベイヤーの配列です。  (X座標,Y座標)= (偶数、偶数)：赤 (偶数、奇数)：緑 (奇数、偶数)：緑 (奇数、奇数)：青 上記以外のベイヤー配列は、カメラの設定を変更するか、RZ/A2M の VIN の機能を使用して対応することが可能です。詳細は解説を参照してください。	

	出力画像	アドレス	: dst で指定 （src と異なるアドレスにしてください）
		幅（ピクセル）	: 入力画像と同じ
		高さ（ピクセル）	: 入力画像と同じ
		フォーマット	: RGB（1 ピクセルあたり 3 バイト）
		データサイズ	: (width) × (height) × 3 バイト
タイル数	2		
分割処理	可		

解説

本機能は、src で指定したアドレスの画像を、ベイヤーフォーマットから RGB 形式の画像に変換し、dst で指定したアドレスに出力します。

本機能は、適応型カラープレーン補間法(ACPI 法)で RGB に変換しています。この手法は特許公開番号 US5629734A の手法です。

本機能では ACPI 法で RGB 変換処理を以下のように行っています。各色及び各座標に分けて説明します。
以下の説明では、入力画像内の座標(x,y)における値を I(x,y)、出力画像内の座標(x,y)における RGB の値を R(x,y)、G(x,y)、B(x,y)と表現します。

1.G 成分の計算

・x＝偶数かつ y＝奇数の場合
・x＝奇数かつ y＝偶数の場合
入力画像のベイヤー配列が G 成分となるため、入力$\mathrm{I}(x,y)$をそのまま使用します。

$$\mathrm{G}(x,y) = \mathrm{I}(x,y)$$

・x＝偶数かつ y＝偶数の場合
・x＝奇数かつ y＝奇数の場合
入力画像のベイヤー配列が R 成分または B 成分となるため、以下のように G(x,y)を計算します。

まず、横方向の変化度合いを示す M と、縦方向の変化度合いを示す N を計算します。

$$\mathrm{M} = |I(x-2,y) + I(x+2,y) - 2 \times I(x,y)| + |I(x+1,y) - I(x-1,y)|$$
$$\mathrm{N} = |I(x,y-2) + I(x,y+2) - 2 \times I(x,y)| + |I(x,y+1) - I(x,y-1)|$$

次に、2 つの変化度合いを比較して、変化度合いが小さい方向で補間値 t を計算します。(変化度合いが同じ場合、補間値 t は両方向の補間値の平均となります。)

① M < N の場合

$$\mathrm{t} = \Big(2 \times \big(I(x-1,y) + I(x+1,y) + I(x,y)\big) - I(x-2,y) - I(x+2,y)\Big) \div 4$$

② M > N の場合

$$\mathrm{t} = \Big(2 \times \big(I(x,y-1) + I(x,y+1) + I(x,y)\big) - I(x,y-2) - I(x,y+2)\Big) \div 4$$

③ M = N の場合

$$\mathrm{t} = \Big(2 \times \big(I(x-1,y) + I(x+1,y) + 2 \times I(x,y) + I(x,y-1) + I(x,y+1)\big) - I(x-2,y) - I(x+2,y) - I(x,y-2) - I(x,y+2)\Big) \div 8$$

補間値 t に対して、小数点以下を切り捨てた値を t′とすると、G(x,y)は下記の値となります。

$$\mathrm{G}(x,y) = \begin{cases} 0, & \mathrm{t}' < 0 \\ 255, & \mathrm{t}' > 255 \\ \mathrm{t}', & 0 \le \mathrm{t}' \le 255 \end{cases}$$

2.R 成分の計算

・x＝偶数かつ y＝偶数の場合

入力画像のベイヤー配列が R 成分となるため、入力$\mathrm{I}(x,y)$をそのまま使用します。

$$\mathrm{R}(x,y) = \mathrm{I}(x,y)$$

・x＝奇数かつ y＝奇数の場合

入力画像のベイヤー配列が B 成分となるため、以下のように R(x,y)を計算します。

まず、斜め(右上～左下)方向の変化度合いを示す M と、斜め(左上～右下)方向の変化度合いを示す N を計算します。

$$\mathrm{M} = |G(x+1,y-1) + G(x-1,y+1) - 2\times G(x,y)| + |I(x-1,y+1) - I(x+1,y-1)|$$
$$\mathrm{N} = |G(x-1,y-1) + G(x+1,y+1) - 2\times G(x,y)| + |I(x+1,y+1) - I(x-1,y-1)|$$

次に、2 つの変化度合いを比較して、変化度合いが小さい方向で補間値 t を計算します。(変化度合いが同じ場合、補間値 t は両方向の補間値の平均となります。)

① M < N の場合

$$\mathrm{t} = \Big(2\times\big(I(x+1,y-1) + I(x-1,y+1) + G(x,y)\big) - G(x+1,y-1) - G(x-1,y+1)\Big) \div 4$$

② M > N の場合

$$\mathrm{t} = \Big(2\times\big(I(x-1,y-1) + I(x+1,y+1) + G(x,y)\big) - G(x-1,y-1) - G(x+1,y+1)\Big) \div 4$$

③ M = N の場合

$$\mathrm{t} = \Big(2\times\big(I(x+1,y-1) + I(x-1,y+1) + 2\times G(x,y) + I(x-1,y-1) + I(x+1,y+1)\big) - G(x+1,y-1) - G(x-1,y+1) - G(x-1,y-1) - G(x+1,y+1)\Big) \div 8$$

補間値 t に対して、小数点以下を切り捨てた値を t′とすると、R(x,y)は下記の値となります。

$$\mathrm{R}(x,y) = \begin{cases} 0, & \mathrm{t}' < 0 \\ 255, & \mathrm{t}' > 255 \\ \mathrm{t}', & 0 \le \mathrm{t}' \le 255 \end{cases}$$

・x＝奇数かつ y＝偶数の場合

入力画像のベイヤー配列が G 成分となるため、入力画像の左右の R 成分と「1.G 成分の計算」で求めた左右の G 成分を考慮して、以下のように R(x,y)を計算します。

$$\mathrm{R}(x,y) = \begin{cases} 0, & \mathrm{M} < \mathrm{N} \\ 255, & \big((M-N) >> 2\big) > 255 \\ (M-N) >> 2, & \text{上記以外} \end{cases}$$
$$\mathrm{M} = 2\times\big(I(x-1,y) + I(x+1,y) + I(x,y)\big)$$
$$\mathrm{N} = \mathrm{G}(x-1,y) + G(x+1,y)$$

・x＝偶数かつ y＝奇数の場合

入力画像のベイヤー配列が G 成分となるため、入力画像の上下の R 成分と「1.G 成分の計算」で求めた上下の G 成分を考慮して、以下のように R(x,y)を計算します。

$$\mathrm{R}(x,y) = \begin{cases} 0, & \mathrm{M} < \mathrm{N} \\ 255, & \big((M-N) >> 2\big) > 255 \\ (M-N) >> 2, & \text{上記以外} \end{cases}$$
$$\mathrm{M} = 2\times\big(I(x,y-1) + I(x,y+1) + I(x,y)\big)$$
$$\mathrm{N} = \mathrm{G}(x,y-1) + G(x,y+1)$$

3.B 成分の計算

・x＝奇数かつ y＝奇数の場合

入力画像のベイヤー配列が B 成分となるため、入力$\mathrm{I}(x,y)$をそのまま使用します。

$$\mathrm{B}(x,y) = \mathrm{I}(x,y)$$

・x＝偶数かつ y＝偶数の場合

入力画像のベイヤー配列が R 成分となるため、以下のように B(x,y)を計算します。

まず、斜め(右上～左下)方向の変化度合いを示す M と、斜め(左上～右下)方向の変化度合いを示す N を計算します。

$$\mathrm{M} = |G(x+1,y-1) + G(x-1,y+1) - 2\times G(x,y)| + |I(x-1,y+1) - I(x+1,y-1)|$$
$$\mathrm{N} = |G(x-1,y-1) + G(x+1,y+1) - 2\times G(x,y)| + |I(x+1,y+1) - I(x-1,y-1)|$$

次に、2 つの変化度合いを比較して、変化度合いが小さい方向で補間値 t を計算します。(変化度合いが同じ場合、補間値 t は両方向の補間値の平均となります。)

① M < N の場合

$$\mathrm{t} = \Big(2\times\big(I(x+1,y-1) + I(x-1,y+1) + G(x,y)\big) - G(x+1,y-1) - G(x-1,y+1)\Big) \div 4$$

② M > N の場合

$$\mathrm{t} = \Big(2\times\big(I(x-1,y-1) + I(x+1,y+1) + G(x,y)\big) - G(x-1,y-1) - G(x+1,y+1)\Big) \div 4$$

③ M = N の場合

$$\mathrm{t} = \Big(2\times\big(I(x+1,y-1) + I(x-1,y+1) + 2\times G(x,y) + I(x-1,y-1) + I(x+1,y+1)\big) - G(x+1,y-1) - G(x-1,y+1) - G(x-1,y-1) - G(x+1,y+1)\Big) \div 8$$

補間値 t に対して、小数点以下を切り捨てた値を t′とすると、B(x,y)は下記の値となります。

$$\mathrm{B}(x,y) = \begin{cases} 0, & \mathrm{t}' < 0 \\ 255, & \mathrm{t}' > 255 \\ \mathrm{t}', & 0 \le \mathrm{t}' \le 255 \end{cases}$$

・x＝偶数かつ y＝奇数の場合

入力画像のベイヤー配列が G 成分となるため、入力画像の左右の B 成分と「1.G 成分の計算」で求めた左右の G 成分を考慮して、以下のように B(x,y)を計算します。

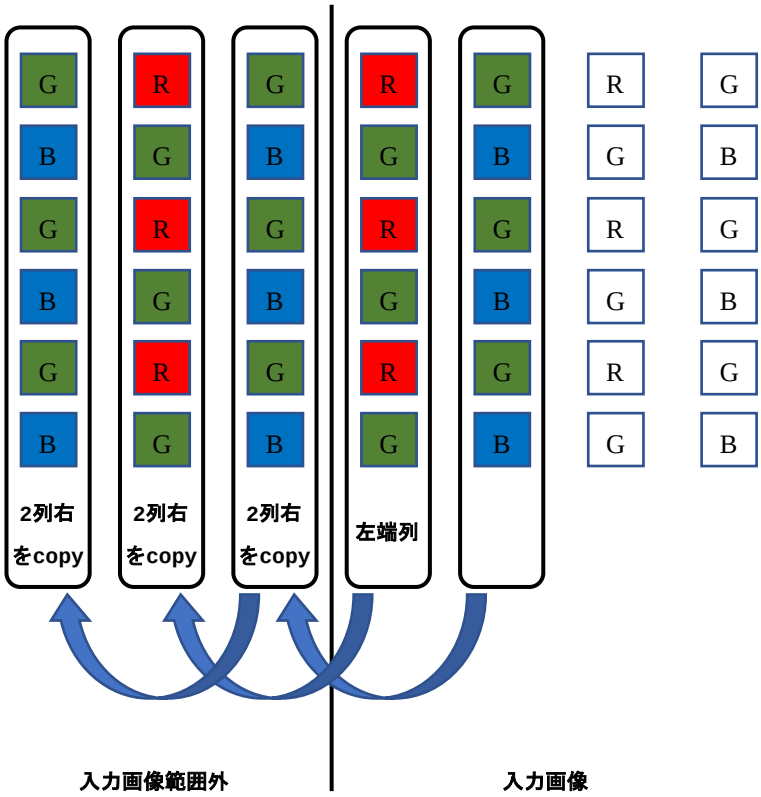
$$\mathrm{B}(x,y) = \begin{cases} 0, & \mathrm{M} < \mathrm{N} \\ 255, & \big((M-N) >> 2\big) > 255 \\ (M-N) >> 2, & \text{上記以外} \end{cases}$$
$$\mathrm{M} = 2\times\big(I(x-1,y) + I(x+1,y) + I(x,y)\big)$$
$$\mathrm{N} = \mathrm{G}(x-1,y) + G(x+1,y)$$

・x＝奇数かつ y＝偶数の場合

入力画像のベイヤー配列が G 成分となるため、入力画像の上下の B 成分と「1.G 成分の計算」で求めた上下の G 成分を考慮して、以下のように B(x,y)を計算します。

$$\mathrm{B}(x,y) = \begin{cases} 0, & \mathrm{M} < \mathrm{N} \\ 255, & \big((M-N) >> 2\big) > 255 \\ (M-N) >> 2, & \text{上記以外} \end{cases}$$
$$\mathrm{M} = 2\times\big(I(x,y-1) + I(x,y+1) + I(x,y)\big)$$
$$\mathrm{N} = \mathrm{G}(x,y-1) + G(x,y+1)$$

画面左右端付近のピクセルを変換する場合には、参照するデータの一部が入力画像の領域外となり参照できないため、その代わりに端 2 列のピクセルの値を参照する、複製境界処理を行います。



top および bottom に 1 を設定した場合は、画面の上下端付近も同様の複製境界処理を行います。入力画像分割を行わない場合は、top と bottom は 1 を設定して下さい。

「入出力詳細」の「入力画像」に示した図以外のベイヤー配列のカメラを使用する場合は、左上が赤色になる位置に画像を切り取ってキャプチャして下さい。画像を切り取る方法は、カメラ側の出力画像範囲をクリップする設定か、もしくは、MIPI カメラを使用する場合であれば、RZ/A2M で入力画像範囲をクリップする方法があります。後者の設定については、RZ/A2M グループ ユーザーズマニュアル ハードウェア編の 48 ビデオインプットモジュール（VIN）を参照頂くか、または、MIPI ドライバのユーザーズマニュアルで、範囲クリップ（前段）の機能を参照してください。

注意	なし

4.9.3 Bayer2RgbColorCorrection

Bayer2 RgbColorCorrection

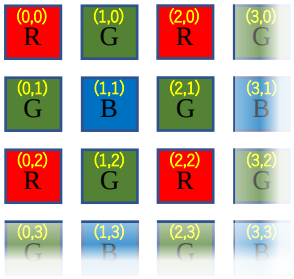
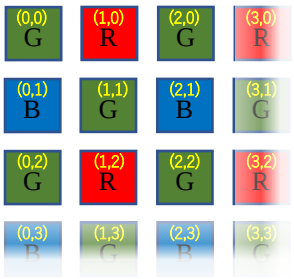
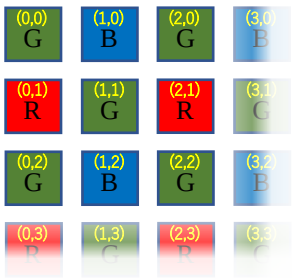
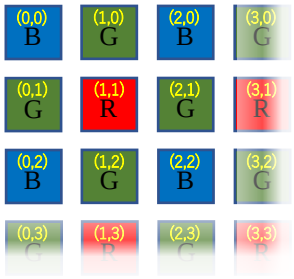
CMOS カメラからの RAW データを RGB カラーへ変換します(色成分補正有)

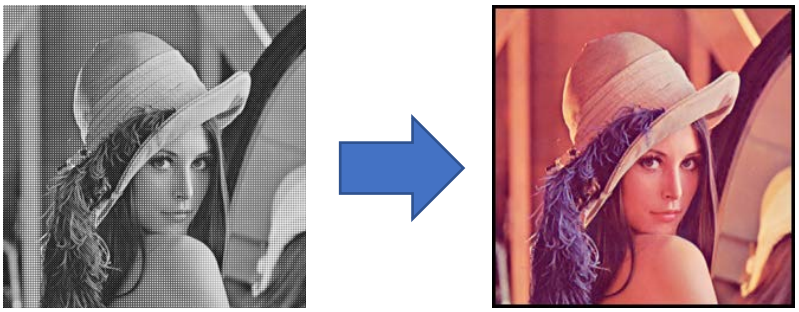
コンフィグレーションデータファイル	r_drp_bayer2rgb_color_correction.dat
対応バージョン	1.01
コンフィグレーションデータサイズ（バイト）	203808
ヘッダファイル	r_drp_bayer2rgb_color_correction.h

パラメータ　構造体名

r_drp_bayer2rgb_color_correction _t

メンバ名	型	説明
src	uint32_t	入力画像のアドレス
dst	uint32_t	出力画像のアドレス
width	uint16_t	画像の幅（ピクセル）
height	uint16_t	画像の高さ（ピクセル）
gain_r	uint16_t	画像のゲイン補正値（R 成分） 上位 4bit が整数部、下位 12bit が小数部
gain_g	uint16_t	画像のゲイン補正値（G 成分） 上位 4bit が整数部、下位 12bit が小数部
gain_b	uint16_t	画像のゲイン補正値（B 成分） 上位 4bit が整数部、下位 12bit が小数部
pattern	uint8_t	入力画像のベイヤー配列のパターンを指定してください。 0:RGGB 1:GRBG 2:GBRG 3:BGGR

入出力詳細	入力画像	アドレス　：src で指定 幅（ピクセル）　：width で指定（16～1280） 高さ（ピクセル）　：height で指定（4～960） データサイズ　：(width) × (height) × 1 バイト ＜フォーマット＞ 入力画像のフォーマットは、下図のような 4 パターンのベイヤー画像です。 RGGB:  (X座標,Y座標)= (偶数、偶数)：赤 (偶数、奇数)：緑 (奇数、偶数)：緑 (奇数、奇数)：青 GRBG:  (X座標,Y座標)= (偶数、偶数)：緑 (偶数、奇数)：青 (奇数、偶数)：赤 (奇数、奇数)：緑 GBRG:  (X座標,Y座標)= (偶数、偶数)：緑 (偶数、奇数)：赤 (奇数、偶数)：青 (奇数、奇数)：緑 BGGR:  (X座標,Y座標)= (偶数、偶数)：青 (偶数、奇数)：緑 (奇数、偶数)：緑 (奇数、奇数)：赤
	出力画像	アドレス　：dst で指定 幅（ピクセル）　：入力画像と同じ 高さ（ピクセル）　：入力画像と同じ フォーマット　：RGB（1 ピクセルあたり 3 バイト） データサイズ　：(width) × (height) × 3 バイト
タイル数	6	
分割処理	不可	

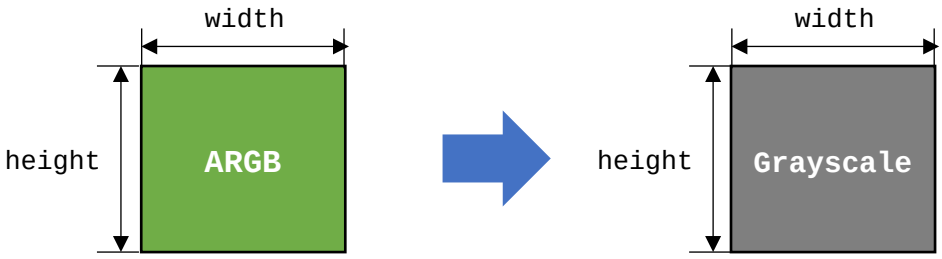
解説	本機能は、src で指定したアドレスの画像に適応型カラープレーン補間(ACPI)法を施して、ベイヤーフォーマットから RGB フォーマットに変換し、dst で指定したアドレスに出力します。 ACPI 法とは補間すべき周囲の画素の線形補間値に高周波数成分を加えることで、鮮鋭なカラー画像を得る手法です。 この手法では、補間値を縦および横の 2 つの方向から求め、処理対象原画素においてより連続性が強いと判断される方向の補間値を採用します。また、足りない画素の色情報をほかの存在する色の情報を用いて導出します。 本機能では、画面端の境界処理は実行しないため、下図のように出力画像の端の上下左右 3 ピクセルには黒いピクセルが出力されます。  入力画像　出力画像 本機能では、パラメータ「gain_*」に補正値を設定することで、ベイヤーから変換した RGB 画像の各成分の画素値を補正できます。ただし、補正値は固定小数点（上位 4bit が整数部、下位 12bit が小数部）であるため、「gain_*」には「実際に補正したい値*4096」の値を設定してください。 本機能は、src と dst に同一アドレスを指定することが可能です。
注意	なし

4.9.4 Argb2Grayscale

Argb2Grayscale

ARGB カラーからグレイスケールへ変換します

コンフィグレーションデータファイル			r_drp_argb2grayscale.dat	
対応バージョン			1.00	
コンフィグレーションデータサイズ（バイト）			14528	
ヘッダファイル			r_drp_argb2grayscale.h	
パラメータ	構造体名			
	r_drp_argb2grayscale_t			
	メンバ名	型	説明	
	src	uint32_t	入力画像のアドレス	
	dst	uint32_t	出力画像のアドレス	
	width	uint16_t	画像の幅（ピクセル）	
	height	uint16_t	画像の高さ（ピクセル）	
入出力詳細	入力画像	アドレス	： src で指定 （dst と異なるアドレスにしてください）	
		幅（ピクセル）	： width で指定（16～1280、2 の整数倍）	
		高さ（ピクセル）	： height で指定（1～960）	
		フォーマット	： ARGB（1 ピクセルあたり 4 バイト）	
		データサイズ	： (width) × (height) × 4 バイト	
	出力画像	アドレス	： dst で指定 （src と異なるアドレスにしてください）	
		幅（ピクセル）	： 入力画像と同じ	
		高さ（ピクセル）	： 入力画像と同じ	
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
タイル数	1			
分割処理	可			
解説	本機能は、src で指定したアドレスの ARGB 形式の画像をグレイスケール形式に変換し、dst で指定したアドレスに出力します。			



本機能の画像フォーマットの変換は、以下の計算式で行います。

Grayscale = (A × 0 + R × 16384 + G × 40960 + B × 8192）÷ 65536

注意	なし

4.9.5 BinarizationFixed

BinarizationFixed

画像を固定閾値(Threshold)で二値画像へ変換します

コンフィグレーションデータファイル		r_drp_binarization_fixed.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		16160	
ヘッダファイル		r_drp_binarization_fixed.h	
パラメータ	構造体名		
	r_drp_binarization_fixed_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	threshold	uint8_t	二値化の閾値（0～255）
入出力詳細	入力画像	アドレス	: src で指定
		幅（ピクセル）	: width で指定（32～1280、8 の整数倍）
		高さ（ピクセル）	: height で指定（1～960）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
	出力画像	アドレス	: dst で指定
		幅（ピクセル）	: 入力画像と同じ
		高さ（ピクセル）	: 入力画像と同じ
		フォーマット	: 8bit グレイスケール（0 or 255） （1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
タイル数	1		
分割処理	可		
解説	本機能は、src で指定したアドレスの画像を二値化し dst で指定したアドレスに出力します。 本機能は、入力データが閾値（threshold メンバ）を超えた場合は 255、閾値以下の場合は 0 を出力します。 本機能は、OpenCV の cv::threshold 関数の引数 Type に THRESH_BINARY を指定した場合の処理内容と同等です。 参考 URL：https://opencv.org/ 本機能は、src と dst に同一アドレスを指定することが可能です。		
注意	なし		

4.9.6 BinarizationAdaptive

BinarizationAdaptive

画像を周囲画像に合わせた動的閾値で二値画像へ変換します

コンフィグレーションデータファイル			r_drp_binarization_adaptive.dat	
対応バージョン			1.00	
コンフィグレーションデータサイズ（バイト）			234560	
ヘッダファイル			r_drp_binarization_adaptive.h	
パラメータ	構造体名			
	r_drp_binarization_adaptive_t			
	メンバ名	型	説明	
	src	uint32_t	入力画像のアドレス	
	dst	uint32_t	出力画像のアドレス	
	width	uint16_t	画像の幅（ピクセル）	
	height	uint16_t	画像の高さ（ピクセル）	
	work	uint32_t	ワークエリアのアドレス	
	range	uint8_t	平均輝度算出時の有効範囲（0～255）	
入出力詳細	入力画像	アドレス	： src で指定	
		幅（ピクセル）	： width で指定（64～1280、32 の整数倍）	
		高さ（ピクセル）	： height で指定（40～960、8 の整数倍）	
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
	出力画像	アドレス	： dst で指定	
		幅（ピクセル）	： 入力画像と同じ	
		高さ（ピクセル）	： 入力画像と同じ	
		フォーマット	： 8bit グレイスケール（0 or 255） （1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
	ワークエリア	アドレス	： work で指定	
		データサイズ	： (((width × height) ÷ 64) + 2)バイト	
		＜内容＞ 平均輝度値を保存するための領域です。平均輝度値については、解説を参照してください。		
タイル数	3			
分割処理	不可			

解説

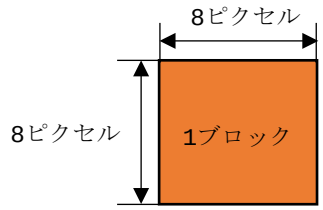
本機能は、src で指定したアドレスの画像を二値化し、dst で指定したアドレスに出力します。

始めに、本機能の二値化処理は、入力画像を 8 × 8 ピクセル単位のブロックに分割し、ブロック毎の平均輝度値を算出します。その後、平均輝度値から閾値を算出して入力画像を二値化します。

平均輝度値の算出方法を説明します。まず、入力画像の左上から 8 × 8 ピクセル単位のブロックに区切ります。そして、ブロック毎に最大輝度値と最小輝度値を探して輝度差を求めます。輝度差が range を超えているブロックは、ブロック内の輝度値の平均値を平均輝度値とします。輝度差が range 以下のブロックは、周囲 3 ブロック（左上、上、左）の平均輝度値から、平均輝度値を求めます。以下に、平均輝度値を求める方法の詳細を示します。

（１）　輝度差がrangeを超えているブロックの場合

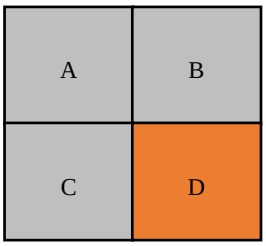
$$\text{平均輝度値} = 8 \times 8\text{ピクセルの輝度値合計} \div 64$$



（２）　輝度差がrange以下のブロックの場合

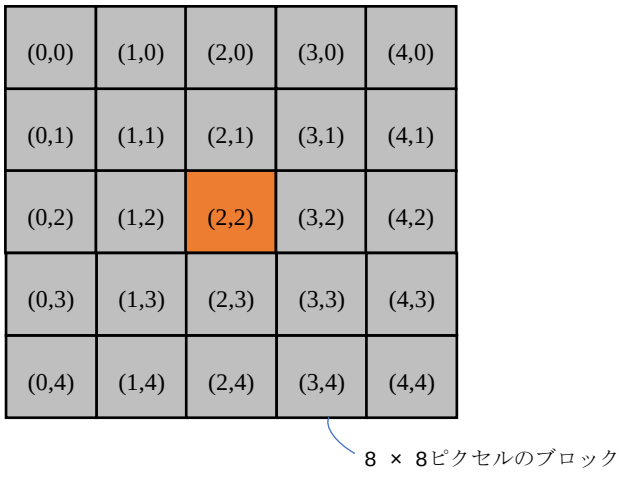
$$\text{平均輝度値} = \left(A\text{の平均輝度値} + \text{B}\text{の平均輝度値} + \left(C\text{の平均輝度値} \times 2\right)\right) \div 4$$

ただし、平均輝度値を算出したいブロック(D) が入力画像の最上端や最左端にあり、周囲3ブロックを確保できない場合は、Dの最小輝度値の1/2の値を平均輝度値とします。



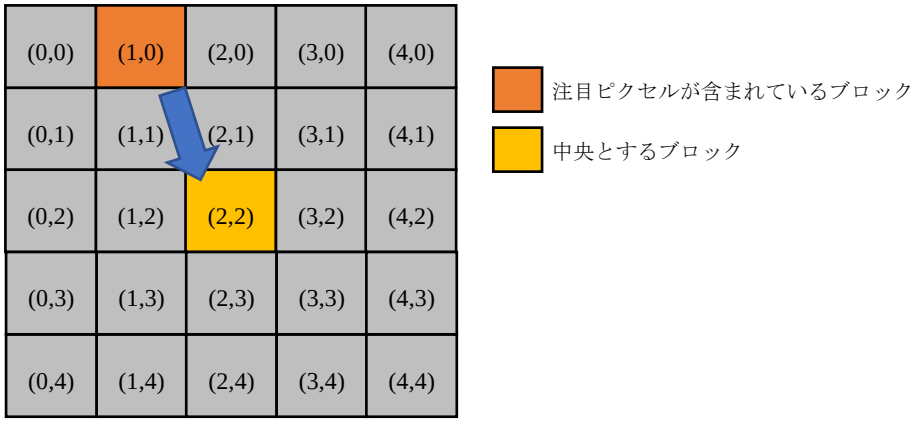
平均輝度値から閾値を算出する方法は、二値化したいピクセル（以後、"注目ピクセル"と表記）が含まれるブロックを中央とした 5 × 5 ブロックに切り出します。この 5 × 5 ブロック全体の平均輝度値から閾値を算出します。以下に、閾値を求める式を示します。

$$\text{閾値} = \left\{(0,0)\text{の平均輝度値} + (1,0)\text{の平均輝度値}+\ldots+(4,4)\text{の平均輝度値}\right\} \div 25$$



ただし、注目ピクセルが含まれているブロックが入力画像の端にあり、5 × 5 ブロックを確保できない場合は、以下のように閾値を算出します。

- 上端 2 ブロック以内、または左端 2 ブロック以内の場合
  5 × 5 ブロックを確保するために、中央ブロックをずらして閾値を算出します。



- 右端 2 ブロック以内の場合
  左隣のブロックの閾値を使用します。
- 下端 2 ブロック以内の場合
  真上のブロックの閾値を使用します。

また、range へ指定する値によって、以下のように二値化の結果が変化します。

入力画像　　range:0　　range:128　　range :255

range の値を小さくすると、濃淡の薄い画像でも白飛びや黒つぶれを抑えた二値化画像を得ることが出来ますが、ノイズの影響を受けやすくなります。range の値を大きくすると、ノイズの影響を受け難くなりますが、白飛びや黒つぶれが発生しやすくなります。range の値は、入力画像（接続するカメラの性能や周囲の環境光など）に合わせて適切な値を設定してください。

本機能は、src と dst に同一アドレスを指定することが可能です。

注意	なし

4.9.7 BinarizationAdaptiveBit

BinarizationAdaptiveBit

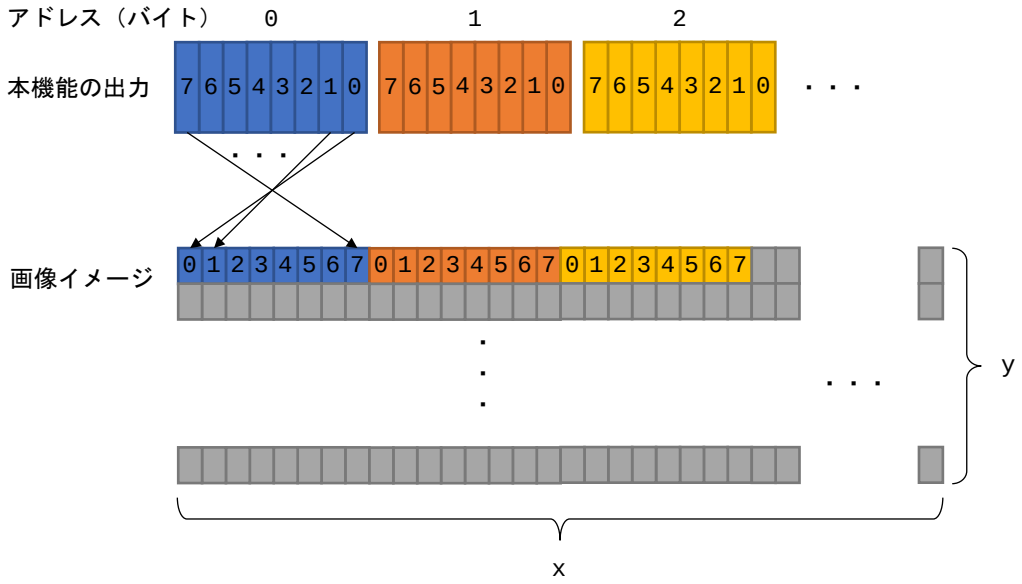
画像を周囲画像に合わせた動的閾値で二値画像へ変換します（ビット出力）

コンフィグレーションデータファイル			r_drp_binarization_adaptive_bit.dat	
対応バージョン			1.00	
コンフィグレーションデータサイズ（バイト）			235712	
ヘッダファイル			r_drp_binarization_adaptive_bit.h	
パラメータ	構造体名			
	r_drp_binarization_adaptive_bit_t			
	メンバ名	型	説明	
	src	uint32_t	入力画像のアドレス	
	dst	uint32_t	出力画像のアドレス	
	width	uint16_t	画像の幅（ピクセル）	
	height	uint16_t	画像の高さ（ピクセル）	
	work	uint32_t	ワークエリアのアドレス	
	range	uint8_t	平均輝度算出時の有効範囲（0～255）	
入出力詳細	入力画像	アドレス	： src で指定	
		幅（ピクセル）	： width で指定（64～1280、32 の整数倍）	
		高さ（ピクセル）	： height で指定（40～960、8 の整数倍）	
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
	出力画像	アドレス	： dst で指定	
		幅（ピクセル）	： 入力画像と同じ	
		高さ（ピクセル）	： 入力画像と同じ	
		フォーマット	： 1 ピクセルあたり 1bit （詳細は解説を参照してください）	
		データサイズ	： (width) × (height) ÷ 8 バイト	
	ワークエリア	アドレス	： work で指定	
		データサイズ	： (((width × height) ÷ 64) + 2)バイト	
		＜内容＞ 平均輝度値を保存するための領域です。平均輝度値については、解説を参照してください。		
タイル数	3			
分割処理	不可			

解説

本機能は、「4.9.6 BinarizationAdaptive」と同じ処理を行います。「4.9.6 BinarizationAdaptive」との違いは処理結果の出力フォーマットのみです。

本機能の出力フォーマットは 1 ピクセルを 1bit で出力します。画像のビットの並びは、x 座標が 0 なら bit0、x 座標が 1 なら bit1、になります。また、白が 0、黒が 1 となります。



また、本機能は、range に 0x18 を指定すると、ZXing で行っている二値化処理（calculateBlackPoints 関数と calculateThresholdForBlock 関数で実現）と同等の処理結果を出力します。

参考 URL：https://github.com/zxing/zxing

本機能は、src と dst に同一アドレスを指定することが可能です。

注意

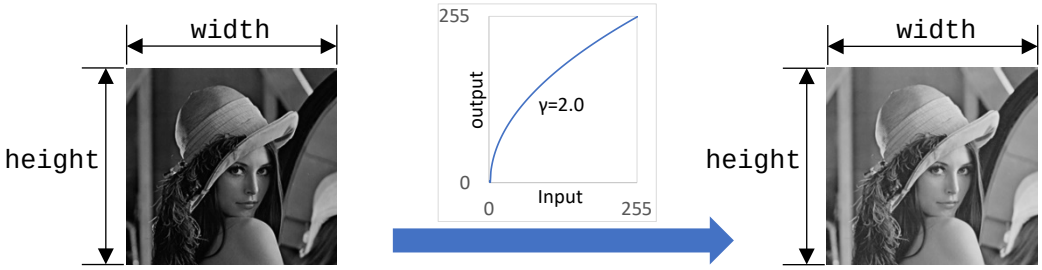
本機能は、「4.9.6 BinarizationAdaptive」との違いは出力フォーマットのみですが、BinarizationAdaptive が 0 を出力するピクセルの場合、BinarizationAdaptiveBit は 1 を出力し、BinarizationAdaptive が 255 を出力するピクセルの場合、BinarizationAdaptiveBit は 0 を出力します。大小関係が逆転しているため、ご注意ください。

4.9.8 GammaCorrection

GammaCorrection

画像全体をガンマ値により補正します

コンフィグレーションデータファイル			r_drp_gamma_correction.dat
対応バージョン			1.01
コンフィグレーションデータサイズ（バイト）			18272
ヘッダファイル			r_drp_gamma_correction.h
パラメータ	構造体名		
	r_drp_gamma_correction_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	table	uint32_t	ガンマ補正に使う LUT アドレス
入出力詳細	入力画像	アドレス	： src で指定
		幅（ピクセル）	： width で指定（16～1280、4 の整数倍）
		高さ（ピクセル）	： height で指定（1～960）
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (width) × (height) × 1 バイト
	出力画像	アドレス	： dst で指定
		幅（ピクセル）	： 入力画像と同じ
		高さ（ピクセル）	： 入力画像と同じ
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (width) × (height) × 1 バイト
タイル数	1		
分割処理	可		
解説	本機能は、src で指定したアドレスの画像にガンマ補正を行い、dst で指定したアドレスに出力します。		



本機能のガンマ補正は、table で指定したルックアップテーブルから補正後の輝度値を取得します。補正前の輝度値を src、補正後の輝度値を dst、ルックアップテーブルを table とすると、補正後の輝度値は以下の式で算出しています。

$$\mathrm{dst} = \mathrm{table}[\mathrm{src}]$$

ルックアップテーブルの大きさは 256 バイト、各値は 0～255 です。ガンマ値を γ とした場合のルックアップテーブルの作成例は、以下のようになります。

$$\mathrm{table}[n] = \left(\frac{n}{255}\right)^{\frac{1}{\gamma}} \times 255 \qquad n = 0\sim255$$

本機能は、src と dst に同一アドレスを指定することが可能です。

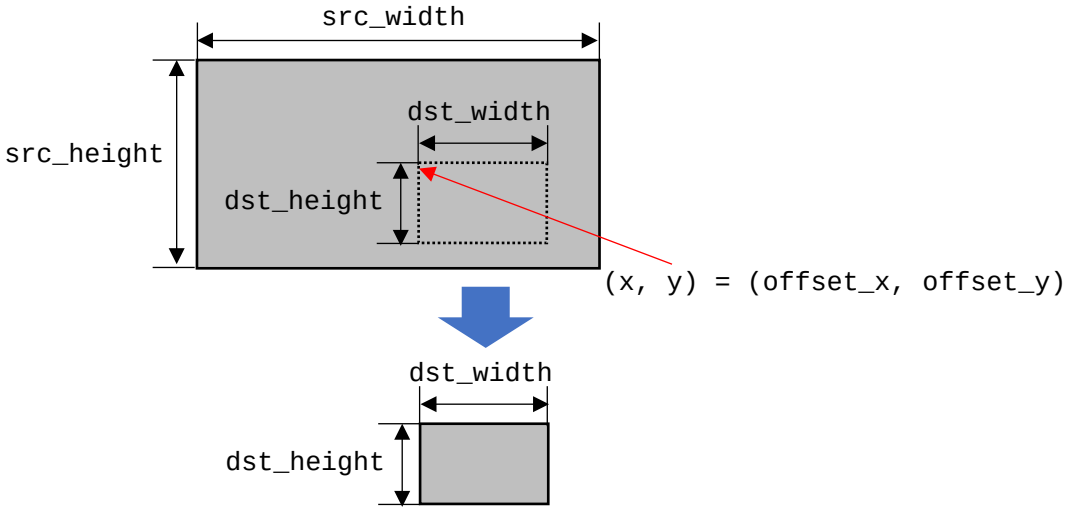
注意	なし

4.9.9 Cropping

Cropping

画像の一部を切り抜きます

コンフィグレーションデータファイル		r_drp_cropping.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		15840	
ヘッダファイル		r_drp_cropping.h	
パラメータ	構造体名		
	r_drp_cropping_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	src_width	uint16_t	入力画像の幅（ピクセル）
	src_height	uint16_t	入力画像の高さ（ピクセル）
	offset_x	uint16_t	入力画像の x 座標
	offset_y	uint16_t	入力画像の y 座標
	dst_width	uint16_t	出力画像の幅（ピクセル）
	dst_height	uint16_t	出力画像の高さ（ピクセル）
入出力詳細	入力画像	アドレス	： src で指定
		幅（ピクセル）	： src_width で指定（8～1280）
		高さ（ピクセル）	： src_height で指定（8～960）
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (src_width) × (src_height) × 1 バイト
	出力画像	アドレス	： dst で指定
		幅（ピクセル）	： dst_width で指定（8～1280、8 の整数倍）
		高さ（ピクセル）	： dst_height で指定（8～960、8 の整数倍）
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (dst_width) × (dst_height) × 1 バイト
タイル数	1		
分割処理	不可		
解説	本機能は、src で指定したアドレスの画像を、指定されたオフセットから指定されたサイズを矩形に切り出し、dst で指定されたアドレスに出力します。		



本機能は、src と dst に同一アドレスを指定することが可能です。

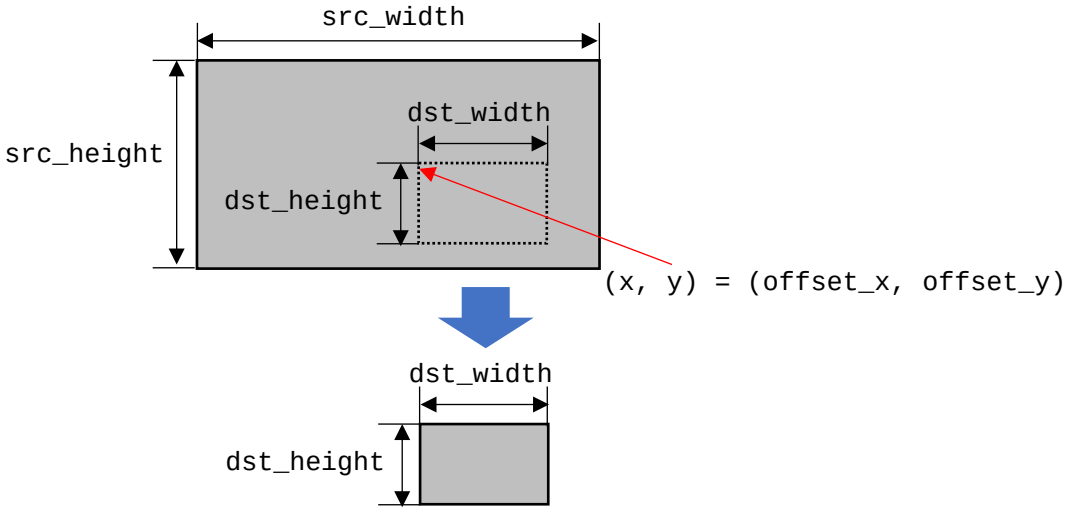
注意	切り出す矩形領域は、入力画像の領域を超えないように引数を指定してください。もし、offset_x＋dst_width が src_width を超えている場合、または、offset_y＋dst_height が src_height を超えている場合は、矩形切り出しを行わずに処理を終了します。

4.9.10 CroppingRgb

CroppingRgb

画像（RGB）の一部を切り抜きます

コンフィグレーションデータファイル		r_drp_cropping_rgb.dat	
対応バージョン		1.01	
コンフィグレーションデータサイズ（バイト）		17440	
ヘッダファイル		r_drp_cropping_rgb.h	
パラメータ	構造体名		
	r_drp_cropping_rgb_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	src_width	uint16_t	入力画像の幅（ピクセル）
	src_height	uint16_t	入力画像の高さ（ピクセル）
	offset_x	uint16_t	入力画像の x 座標
	offset_y	uint16_t	入力画像の y 座標
	dst_width	uint16_t	出力画像の幅（ピクセル）
	dst_height	uint16_t	出力画像の高さ（ピクセル）
入出力詳細	入力画像	アドレス	： src で指定
		幅（ピクセル）	： src_width で指定（8～1280）
		高さ（ピクセル）	： src_height で指定（8～960）
		フォーマット	： RGB（1 ピクセルあたり 3 バイト）
		データサイズ	： (src_width) × (src_height) × 3 バイト
	出力画像	アドレス	： dst で指定
		幅（ピクセル）	： dst_width で指定（8～1280、8 の整数倍）
		高さ（ピクセル）	： dst_height で指定（8～960、8 の整数倍）
		フォーマット	： RGB（1 ピクセルあたり 3 バイト）
		データサイズ	： (dst_width) × (dst_height) × 3 バイト
タイル数	1		
分割処理	不可		
解説	本機能は、src で指定したアドレスの画像を、指定されたオフセットから指定されたサイズを矩形に切り出し、dst で指定されたアドレスに出力します。		



本機能は、src と dst に同一アドレスを指定することが可能です。

注意	切り出す矩形領域は、入力画像の領域を超えないように引数を指定してください。 offset_x＋dst_width が src_width を超えている場合、または、offset_y＋dst_height が src_height を超えている場合は、矩形切り出しを行わずに処理を終了します。

4.9.11 ResizeBilinearFixed

ResizeBilinearFixed

画像のサイズを変更します(バイリニア法 倍率:2^n 倍)

コンフィグレーションデータファイル			r_drp_resize_bilinear_fixed.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			115744
ヘッダファイル			r_drp_resize_bilinear_fixed.h
パラメータ	構造体名		
	r_drp_resize_bilinear_fixed_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	src_width	uint16_t	入力画像の横幅（ピクセル）
	src_height	uint16_t	入力画像の縦幅（ピクセル）
	fx	uint8_t	水平方向のスケールファクタ 拡大・縮小率は以下のようになります。出力画像の幅が 8 ピクセル以上になるようにしてください。

設定値	拡大・縮小率
0x80	0.125（1/8）
0x40	0.25（1/4）
0x20	0.5（1/2）
0x10	1 倍（等倍）
0x08	2 倍
0x04	4 倍
0x02	8 倍
0x01	16 倍

	fy	uint8_t	垂直方向のスケールファクタ fx と設定値は同じです。
入出力詳細	入力画像	アドレス	： src で指定 （dst と異なるアドレスにしてください）
		幅（ピクセル）	： src_width で指定（128～1280）
		高さ（ピクセル）	： src_height で指定（8～960）
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (src_width) × (src_height) × 1 バイト
	出力画像	アドレス	： dst で指定 （src と異なるアドレスにしてください）
		幅（ピクセル）	： src_width ×（水平方向の拡大・縮小率）
		高さ（ピクセル）	： src_height ×（垂直方向の拡大・縮小率）
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (出力画像の幅) × (出力画像の高さ) × 1 バイト
タイル数	4		
分割処理	不可		
解説	本機能は、src で指定したアドレスの画像を指定された倍率で拡大または縮小を行い、dst で指定されたアドレスに出力します。 画像の拡大、縮小を行う場合、ピクセルの補間・間引きが必要になりますが、本機能ではバイリニア法を使用しています。 バイリニア法とは、出力画像の注目ピクセルに対し、対応する入力画像の位置の周辺 2 × 2 ピクセルを用いて、線形補間を行う補間法です。 本機能は OpenCV の cv::resize 関数の引数 dsize に 0、fx と fy に 0.125～16 までの拡大・縮小率、interpolation に INTER_LINEAR を指定した場合と同等の結果が得られます。 参考 URL： https://opencv.org/		
注意	なし		

4.9.12 ResizeBilinearFixedRgb

ResizeBilinearFixedRgb

画像のサイズを変更します(バイリニア法 倍率:2^n 倍)

コンフィグレーションデータファイル			r_drp_resize_bilinear_fixed_rgb.dat
対応バージョン			1.01
コンフィグレーションデータサイズ（バイト）			189600
ヘッダファイル			r_drp_resize_bilinear_fixed_rgb.h
パラメータ	構造体名		
	r_drp_resize_bilinear_fixed_rgb_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	src_width	uint16_t	入力画像の横幅（ピクセル）
	src_height	uint16_t	入力画像の縦幅（ピクセル）
	fx	uint8_t	水平方向のスケールファクタ 拡大・縮小率は以下のようになります。出力画像の幅が 8 ピクセル以上になるようにしてください。

設定値	拡大・縮小率
0x80	0.125（1/8）
0x40	0.25（1/4）
0x20	0.5（1/2）
0x10	1 倍（等倍）
0x08	2 倍
0x04	4 倍
0x02	8 倍
0x01	16 倍

	fy	uint8_t	垂直方向のスケールファクタ fx と設定値は同じです。
入出力詳細	入力画像	アドレス	： src で指定 （dst と異なるアドレスにしてください）
		幅（ピクセル）	： src_width で指定（128～1280）
		高さ（ピクセル）	： src_height で指定（8～960）
		フォーマット	： RGB（1 ピクセルあたり 3 バイト）
		データサイズ	： (src_width) × (src_height) × 3 バイト
	出力画像	アドレス	： dst で指定 （src と異なるアドレスにしてください）
		幅（ピクセル）	： src_width ×（水平方向の拡大・縮小率）
		高さ（ピクセル）	： src_height ×（垂直方向の拡大・縮小率）
		フォーマット	：RGB（1 ピクセルあたり 3 バイト）
		データサイズ	： (出力画像の幅) × (出力画像の高さ) ×3 バイト
タイル数	6		
分割処理	不可		
解説	本機能は、src で指定したアドレスの画像を指定された倍率で拡大または縮小を行い、dst で指定されたアドレスに出力します。 画像の拡大、縮小を行う場合、ピクセルの補間・間引きが必要になりますが、本機能ではバイリニア法を使用しています。 バイリニア法とは、出力画像の注目ピクセルに対し、対応する入力画像の位置の周辺 2 × 2 ピクセルを用いて、線形補間を行う補間法です。 本機能は OpenCV の cv::resize 関数の引数 dsize に 0、fx と fy に 0.125～16 までの拡大・縮小率、interpolation に INTER_LINEAR を指定した場合と同等の結果が得られます。 参考 URL： https://opencv.org/		
注意	なし		

4.9.13 ResizeBilinear

ResizeBilinear

画像のサイズを変更します(バイリニア法 倍率:任意)

コンフィグレーションデータファイル			r_drp_resize_bilinear.dat	
対応バージョン			1.00	
コンフィグレーションデータサイズ（バイト）			390880	
ヘッダファイル			r_drp_resize_bilinear.h	
パラメータ	構造体名			
	r_drp_resize_bilinear_t			
	メンバ名	型	説明	
	src	uint32_t	入力画像のアドレス	
	dst	uint32_t	出力画像のアドレス	
	src_width	uint16_t	入力画像の横幅（ピクセル）	
	src_height	uint16_t	入力画像の縦幅（ピクセル）	
	dst_width	uint16_t	出力画像の横幅（ピクセル）	
	dst_height	uint16_t	出力画像の縦幅（ピクセル）	
入出力詳細	入力画像	アドレス	: src で指定 （dst と異なるアドレスにしてください）	
		幅（ピクセル）	: src_width で指定（32～1280）	
		高さ（ピクセル）	: src_height で指定（8～960）	
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	: (src_width) × (src_height) × 1 バイト	
	出力画像	アドレス	: dst で指定 （src と異なるアドレスにしてください）	
		幅（ピクセル）	: dst_width で指定（32～1280）	
		高さ（ピクセル）	: dst_height で指定（8～960）	
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	: (dst_width) × (dst_height) × 1 バイト	
タイル数	6			
分割処理	不可			

解説

本機能は、src で指定したアドレスの画像を拡大または縮小を行い、dst で指定されたアドレスに出力します。

画像の拡大、縮小を行う場合、ピクセルの補間・間引きが必要になりますが、本機能ではバイリニア法を使用しています。

バイリニア法とは、出力画像の注目ピクセルに対し、対応する入力画像の位置の周辺 2 × 2 ピクセルを用いて、線形補間を行う補間法です。本機能では、バイリニア法を下記のように計算します。

出力画像内の座標(dx,dy)に対応する入力画像内の座標を(sx,sy)とすると、sx、sy は下記の式で表されます。

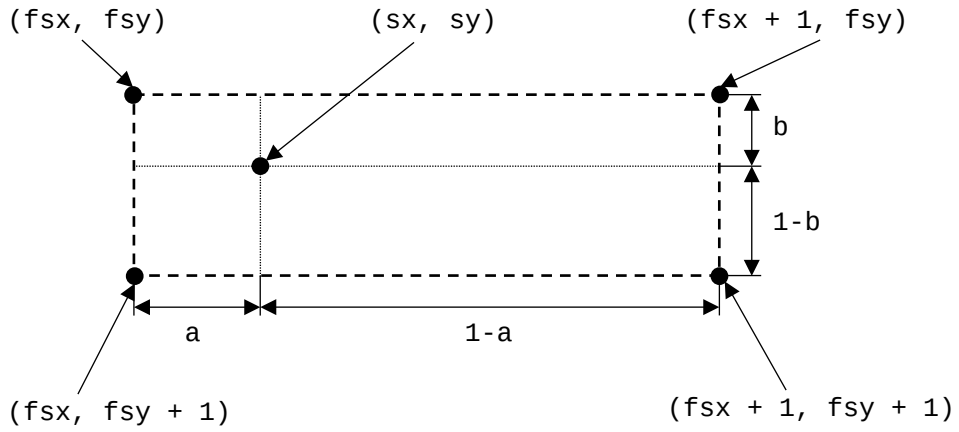
$$sx = (dx + 0.5) \times src\_width \div dst\_width - 0.5$$
$$sy = (dy + 0.5) \times src\_height \div dst\_height - 0.5$$

fsx=Floor(sx)、fsy=Floor(sy)とすると、(sx,sy)の周辺 2 × 2 ピクセルの座標は、(fsx,fsy)、(fsx+1,fsy)、(fsx,fsy+1)、(fsx+1,fsy+1)となります。

座標(x,y)の入力画像内の輝度値を src(x,y)、出力画像内の輝度値を dst(x,y)とすると、dst(dx,dy)は下記の式で表されます。

$$dst(dx, dy) = (1 - b) \times (1 - a) \times src(fsx, fsy) + (1 - b) \times a \times src(fsx + 1, fsy)$$
$$+ b \times (1 - a) \times src(fsx, fsy + 1) + b \times a \times src(fsx + 1, fsy + 1)$$

ただし、a = sx – fsx、b = sy – fsy



本機能は OpenCV の cv::resize 関数の引数 dsize.width に dst_width、dsize.height に dst_height、interpolation に INTER_LINEAR を指定した場合と同等の結果が得られます。

参考 URL： https://opencv.org/

注意　なし

4.9.14 ResizeNearest

ResizeNearest

画像のサイズを変更します(ニアレストネイバー法 倍率:任意)

コンフィグレーションデータファイル			r_drp_resize_nearest.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			294912
ヘッダファイル			r_drp_resize_nearest.h
パラメータ	構造体名		
	r_drp_resize_nearest_t		

メンバ名	型	説明
src	uint32_t	入力画像のアドレス
dst	uint32_t	出力画像のアドレス
src_width	uint16_t	入力画像の横幅（ピクセル）
src_height	uint16_t	入力画像の縦幅（ピクセル）
dst_width	uint16_t	出力画像の横幅（ピクセル）
dst_height	uint16_t	出力画像の縦幅（ピクセル）

入出力詳細	入力画像	アドレス	： src で指定 （dst と異なるアドレスにしてください）
		幅（ピクセル）	： src_width で指定（32～1280）
		高さ（ピクセル）	： src_height で指定（8～960）
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (src_width) × (src_height) × 1 バイト
	出力画像	アドレス	： dst で指定 （src と異なるアドレスにしてください）
		幅（ピクセル）	： dst_width で指定（32～1280）
		高さ（ピクセル）	： dst_height で指定（8～960）
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (dst_width) × (dst_height) × 1 バイト
タイル数	6		
分割処理	不可		

解説

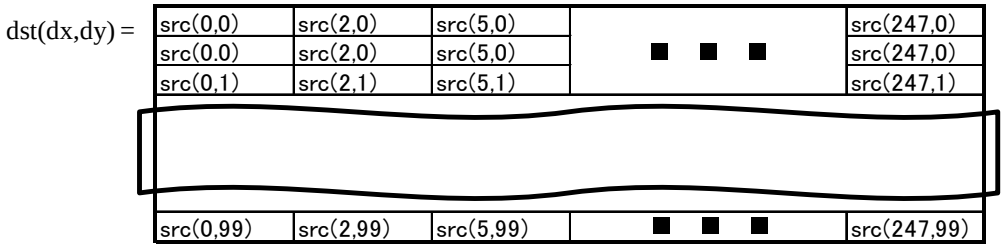
本機能は、src で指定したアドレスの画像を拡大または縮小を行い、dst で指定されたアドレスに出力します。

出力画像内の座標(dx,dy)の輝度値 dst(dx,dy)は、入力画像内の座標(x,y)の輝度値を src(x,y)と表現すると、下記の式で表されます。

$$dst(dx, dy) = src(dx \times src\_width \div dst\_width, dy \times src\_height \div dst\_height)$$

【座標値は小数点以下切り捨て】

下図に、例として入力画像サイズ 250×100、出力画像サイズ 100×200 の時の出力画像を示します。



本機能は OpenCV の cv::resize 関数の引数 dsize.width に dst_width、dsize.height に dst_height、interpolation に INTER_NEAREST を指定した場合と同等の結果が得られます。

参考 URL： https://opencv.org/

注意　なし

4.9.15 ImageRotate

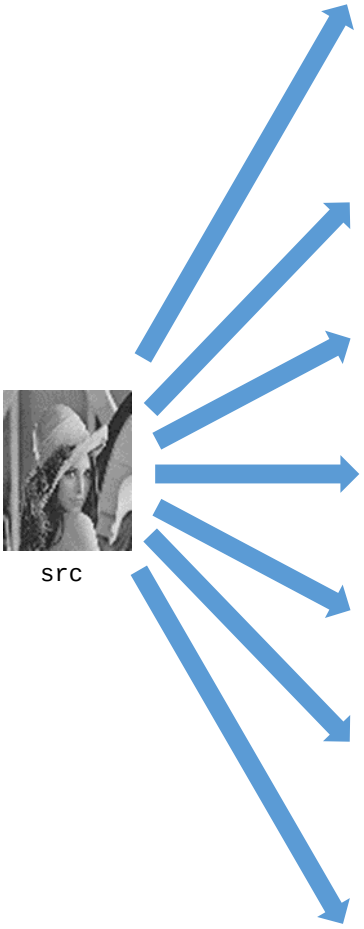
ImageRotate

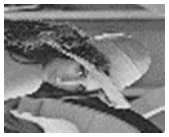
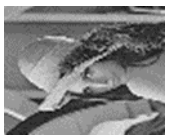
画像を回転します

コンフィグレーションデータファイル			r_drp_image_rotate.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			56224
ヘッダファイル			r_drp_image_rotate.h
パラメータ	構造体名		
	r_drp_image_rotate_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	src_width	uint16_t	入力画像の幅（ピクセル）
	src_height	uint16_t	入力画像の高さ（ピクセル）
	dst_stride	uint16_t	出力画像のストライド値（0～1920） 0 を設定すると、出力画像の各ラインが連続したアドレスに出力されます。 0 以外を設定すると、出力画像の各ラインが本パラメータ値分の間隔で出力されます。 0、または出力画像の幅より大きな値を設定してください。最大値は 1920 です。 詳細は解説を参照してください。
	mode	uint8_t	1:時計回り方向 90° 回転 2:時計回り方向 180° 回転 3:時計回り方向 270° 回転 4:左右反転 5:左右反転後、時計回り方向 90° 回転 6:左右反転後、時計回り方向 180° 回転 7:左右反転後、時計回り方向 270° 回転 詳細は解説を参照してください。
入出力詳細	入力画像	アドレス	: src で指定 （dst と異なるアドレスにしてください）
		幅（ピクセル）	: src_width で指定（16～1280）
		高さ（ピクセル）	: src_height で指定（8～960、2 の整数倍）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (src_width) × (src_height) × 1 バイト
	出力画像	アドレス	: dst で指定 （src と異なるアドレスにしてください）
		幅（ピクセル）	: dst_stride が 0 の場合 ・mode＝2,4,6 の時、src_width と同じ ・mode＝1,3,5,7 の時、src_height と同じ dst_stride が 0 以外の場合 ・dst_stride と同じ
		高さ（ピクセル）	: mode＝2,4,6 の時、src_height と同じ mode＝1,3,5,7 の時、src_width と同じ
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: dst_stride が 0 の場合 (src_width) × (src_height) × 1 バイト dst_stride が 0 以外の場合 ・mode=2,4,6 の時 (dst_stride) × (src_height) × 1 バイト ・mode=1,3,5,7 の時 (src_width) × (dst_stride) × 1 バイト
タイル数	1		
分割処理	可 詳細は解説を参照してください。		

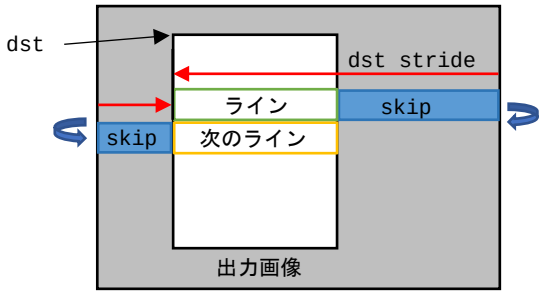
解説

本機能は、src で指定したアドレスの画像に mode で指定された回転及び反転を行い、dst で指定したアドレスに出力します。

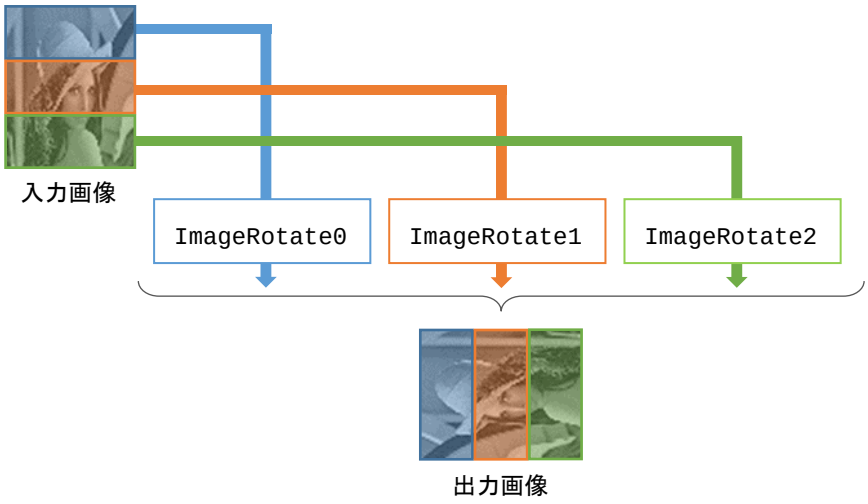


mode	dst
1	
2	
3	
4	
5	
6	
7	

通常は、dst_stride に 0 を設定してください。
本機能は、大きな画像の部分矩形領域に画像を出力することが可能です。その場合は、dst_stride に画像出力時の各ラインから次のラインまでのピクセル数を設定してください。



本機能は、分割処理を行う事が出来ます。
3 並列処理を行う例を以下に示します。
入力データを 3 つの領域に分割し、ImageRotate0、ImageRotate1、ImageRotate2 に対して、それぞれ所定の src、dst、src_height を指定します。src_width、dst_stride、mode は同じ設定として下さい。
mode に 1、3、5、7 を設定する場合、3 つ合わせた出力画像の幅が 3 つの src_height を加算した値になるため、dst_stride には 3 つの src_height を加算した値を設定してください。mode がそれ以外の場合は、3 つ合わせた出力画像の幅が src_width と同じ値になるため、dst_stride には 0 を設定してください。



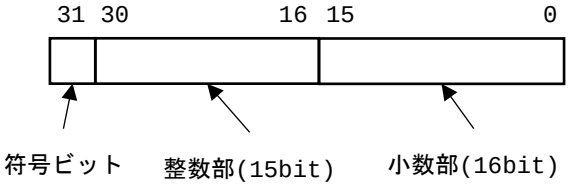
本機能の mode=1 は、OpenCV にて cv::flip(src, tmp, 0);cv::transpose(tmp, dst);
を実施した場合と同等の結果が得られます。
本機能の mode=2 は、OpenCV にて cv::flip(src, dst, -1);
を実施した場合と同等の結果が得られます。
本機能の mode=3 は、OpenCV にて cv::flip(src, tmp, 1);cv::transpose(tmp, dst);
を実施した場合と同等の結果が得られます。
本機能の mode=4 は、OpenCV にて cv::flip(src, dst, 1);
を実施した場合と同等の結果が得られます。
本機能の mode=5 は、OpenCV にて cv::flip(src, tmp, -1);cv::transpose(tmp, dst);
を実施した場合と同等の結果が得られます。
本機能の mode=6 は、OpenCV にて cv::flip(src, dst, 0);
を実施した場合と同等の結果が得られます。
本機能の mode=7 は、OpenCV にて cv::transpose(src, dst);
を実施した場合と同等の結果が得られます。
参考 URL： https://opencv.org/

注意　なし

4.9.16 Affine

Affine

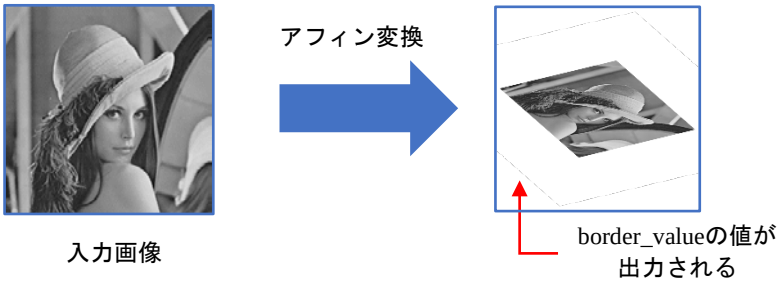
画像の平行移動、線形変換を行います

コンフィグレーションデータファイル			r_drp_affine.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			728448
ヘッダファイル			r_drp_affine.h
パラメータ	構造体名		
	r_drp_affine_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	src_width	uint16_t	入力画像の幅（ピクセル）
	src_height	uint16_t	入力画像の高さ（ピクセル）
	dst_width	uint16_t	出力画像の幅（ピクセル）
	dst_height	uint16_t	出力画像の高さ（ピクセル）
	m_11	int32_t	変換行列における 1 行 1 列目の要素を固定小数化した値 固定小数のフォーマットは下図の通りです。  表現範囲(-32768～+32767.9999847412109375) （詳細は解説を参照してください）
	m_12	int32_t	変換行列における 1 行 2 列目の要素を固定小数化した値 固定小数のフォーマットは m_11 と同じです。 （詳細は解説を参照してください）
	m_13	int32_t	変換行列における 1 行 3 列目の要素を固定小数化した値 固定小数のフォーマットは m_11 と同じです。 （詳細は解説を参照してください）
	m_21	int32_t	変換行列における 2 行 1 列目の要素を固定小数化した値 固定小数のフォーマットは m_11 と同じです。 （詳細は解説を参照してください）
	m_22	int32_t	変換行列における 2 行 2 列目の要素を固定小数化した値 固定小数のフォーマットは m_11 と同じです。 （詳細は解説を参照してください）
	m_23	int32_t	変換行列における 2 行 3 列目の要素を固定小数化した値 固定小数のフォーマットは m_11 と同じです。 （詳細は解説を参照してください）
	border_value	uint8_t	参照する入力画像が範囲外となったときの出力値
入出力詳細	入力画像	アドレス	： src で指定 （dst と異なるアドレスにしてください）
		幅（ピクセル）	： src_width で指定（32～1280）
		高さ（ピクセル）	： src_height で指定（8～960）
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (src_width) × (src_height) × 1 バイト

	出力画像	アドレス	: dst で指定 （src と異なるアドレスにしてください）
		幅（ピクセル）	: dst_width で指定（32～1280）
		高さ（ピクセル）	: dst_height で指定（8～960）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (dst_width) × (dst_height) × 1 バイト
タイル数	6		
分割処理	不可		

解説

本機能は、src で指定したアドレスの画像に対してアフィン変換を行い、dst で指定したアドレスに出力します。



変換行列 M を、

$$M = \begin{pmatrix} m\_11 & m\_12 & m\_13 \\ m\_21 & m\_22 & m\_23 \\ 0 & 0 & 1 \end{pmatrix}$$

と定義すると、アフィン変換によって入力画像の座標(sx,sy)は、以下の式で表される座標(dx,dy)に写像されます。

$$\begin{pmatrix} dx \\ dy \\ 1 \end{pmatrix} = M \begin{pmatrix} sx \\ sy \\ 1 \end{pmatrix}$$

例として、画像回転後、拡大縮小を行う場合、入力画像を回転する際の中心座標を(cx,cy)、回転角度を θ (反時計回り方向が正方向)、画像拡大率を s とすると変換行列 M は、

$$M = \begin{pmatrix} s \times \cos\theta & s \times \sin\theta & (1 - s \times \cos\theta) \times cx - s \times \sin\theta \times cy \\ -s \times \sin\theta & s \times \cos\theta & s \times \sin\theta \times cx + (1 - s \times \cos\theta) \times cy \\ 0 & 0 & 1 \end{pmatrix}$$

となります。
また、OpenCV の関数 **cv::getRotationMatrix2D**、**cv::getAffineTransform** 等を使用して、下記の枠線部分の 2×3 変換行列を生成することができます。

$$M = \begin{pmatrix} m\_11 & m\_12 & m\_13 \\ m\_21 & m\_22 & m\_23 \\ 0 & 0 & 1 \end{pmatrix}$$

関数 **cv::getRotationMatrix2D** は、画像の回転を行う変換行列を生成します。
関数 **cv::getAffineTransform** は、入出力画像の 3 点を指定しそれぞれの点を写像する変換行列を生成します。
参考 URL： https://opencv.org/

本機能では、出力画像の座標から逆変換で入力画像の座標を計算し、アフィン変換を行います。行列 M の逆行列を M^{-1} とすると、逆変換は以下のように計算できます。

$$\begin{pmatrix} sx \\ sy \\ 1 \end{pmatrix} = M^{-1} \begin{pmatrix} dx \\ dy \\ 1 \end{pmatrix}$$

本機能では、出力画像を計算する際、周囲 4 ピクセルを参照するバイリニア法を用いて補間します。バイリニア法の詳細は、ResizeBilinear の章を参照してください。

本機能は、OpenCV の cv::warpAffine 関数の引数 M にアフィン変換の内容に応じた変換行列、dsize.width に dst_width、dsize.height に dst_height、flags に INTER_LINEAR、borderMode に BORDER_CONSTANT、borderValue に border_value を指定した場合と同等の結果が得られます。
参考 URL： https://opencv.org/

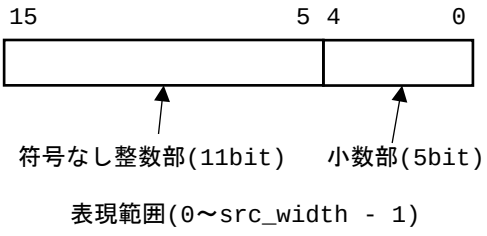
注意　変換行列 M のパラメータについて、「m_11×m_22＝m_12×m_21」となるように設定した場合、逆行列 M^{-1} が計算できないため、出力画像領域の全てにおいて border_value が出力されます。

4.9.17 Remap

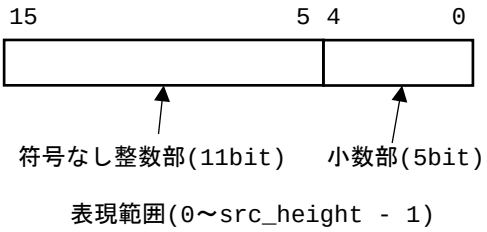
Remap

X,Y 座標値マップデータを用いて画像変換を行います

コンフィグレーションデータファイル		r_drp_remap.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		219584	
ヘッダファイル		r_drp_remap.h	
パラメータ	構造体名		
	r_drp_remap_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	src_width	uint16_t	入力画像の幅（ピクセル）
	src_height	uint16_t	入力画像の高さ（ピクセル）
	dst_width	uint16_t	出力画像の幅（ピクセル）
	dst_height	uint16_t	出力画像の高さ（ピクセル）
	mapx	uint32_t	X 座標値マップのアドレス
	mapy	uint32_t	Y 座標値マップのアドレス
	border_value	uint8_t	X 座標値マップ及び Y 座標値マップから取得した入力画像上の座標が、入力画像の範囲外となった場合の出力値
入出力詳細	入力画像	アドレス	: src で指定
		幅（ピクセル）	: src_width で指定（8～1920）
		高さ（ピクセル）	: src_height で指定（8～1080）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (src_width) × (src_height) × 1 バイト
	X 座標値マップ（入力）	アドレス	: mapx で指定
		幅（ピクセル）	: dst_width で指定（8～1920、2 の整数倍）
		高さ（ピクセル）	: dst_height で指定（8～1080）
		フォーマット	: 入力画像上の X 座標値を、16bit 符号なし固定小数点数で表現。表現範囲外の値の場合、出力画像上のデータは border_value となる
		データサイズ	: (dst_width) × (dst_height) × 2 バイト



	Y 座標値マップ（入力）	アドレス	: mapy で指定
		幅（ピクセル）	: dst_width で指定（8～1920、2 の整数倍）
		高さ（ピクセル）	: dst_height で指定（8～1080）
		フォーマット	: 入力画像上の Y 座標値を、16bit 符号なし固定小数点数で表現。表現範囲外の値の場合、出力画像上のデータは border_value となる



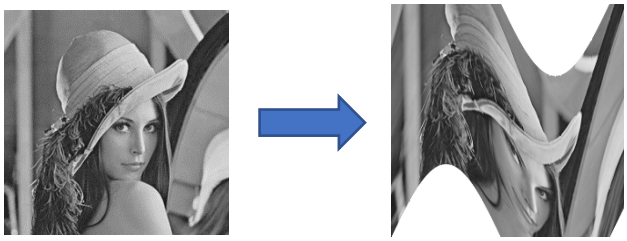
表現範囲(0～src_height - 1)

		データサイズ	: (dst_width) × (dst_height) × 2 バイト
	出力画像	アドレス	: dst で指定
		幅（ピクセル）	: dst_width で指定（8～1920、2 の整数倍）
		高さ（ピクセル）	: dst_height で指定（8～1080）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (dst_width) × (dst_height) × 1 バイト
タイル数	6		
分割処理	不可		

解説

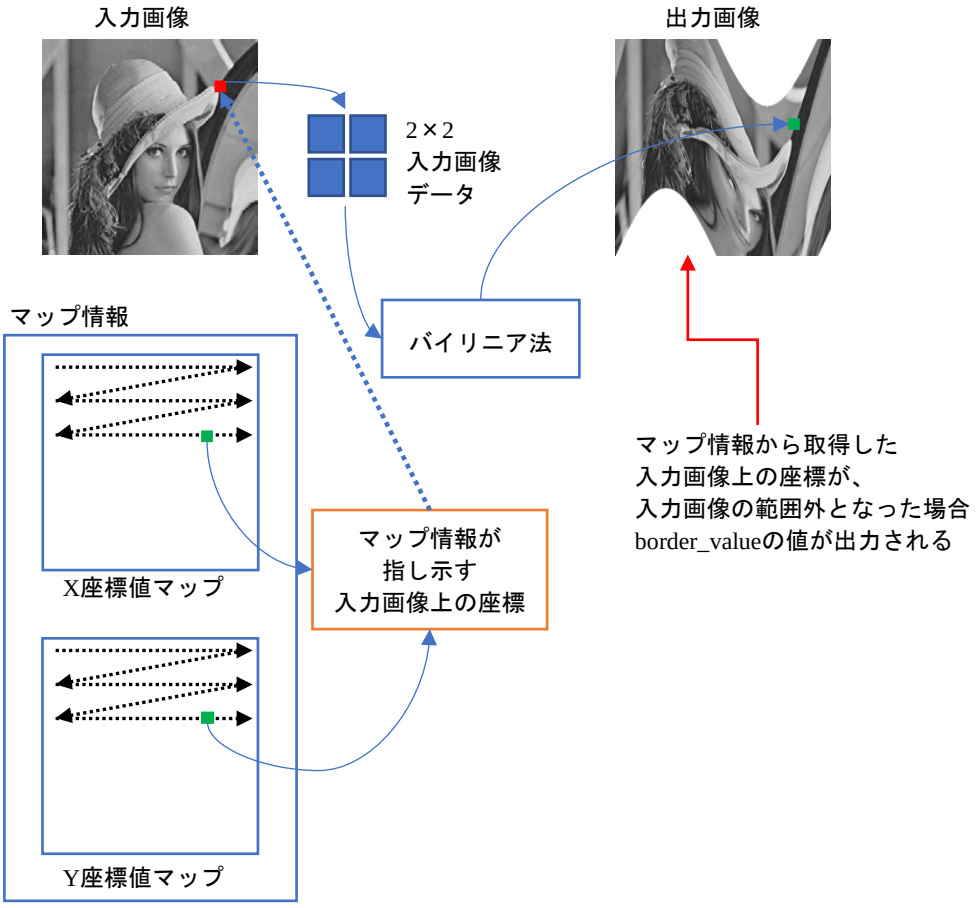
本機能は、src で指定したアドレスの画像に対して、mapx で指定したアドレスの入力画像の X 座標値データと、mapy で指定したアドレスの入力画像の Y 座標値データとを用いて画像変換を行い、dst で指定したアドレスに出力します。

本機能の画像変換は、縮小や上下左右反転等自由に画像を変形することができ、また、カメラ等のレンズ歪み補正にも使用することができます。



本機能を使用した画像変換の例

画像変換は、マップ情報を元に入力画像から対応する画像データをバイリニア法で補間し、その補間データをマップ情報の座標に対応した出力画像の座標に書き込むことで行います。
出力画像を計算する際、X 座標値マップと Y 座標値マップから取得した入力画像の座標の周囲 4 ピクセルを参照するバイリニア法を用いて補間しています。バイリニア法の詳細は、「4.9.13 ResizeBilinear」の章を参照してください。
また、入力画像上の座標が範囲外の場合は、border_value の値が適応されます。



マップ情報の作成方法について説明します。

例えば、カメラで撮影した画像に対する歪み補正等の画像変換を行うために必要な「X 座標値マップ」と「Y 座標値マップ」は、OpenCV の関数 **cv::initUndistortRectifyMap** を使用して生成することができます。**initUndistortRectifyMap** では、32bit の float 型の「X 座標値マップ」と「Y 座標値マップ」を生成するため、その 32bit の float 型の各座標値データを、符号なし整数部 11bit 小数部 5bit の固定小数点数に変換する必要があります。

initUndistortRectifyMap 以外の 32bit の float 型の「X 座標値マップ」と「Y 座標値マップ」を作成する例については、OpenCV 公式ホームページの Remapping のチュートリアルを参照してください。

OpenCV 公式ホームページ URL：https://opencv.org/
（Tutorials をクリックした後、「Remapping」でサーチ）

・**initUndistortRectifyMap** を使用してマップ情報を作成する場合
initUndistortRectifyMap の引数「m1type」には、必ず「CV_32FC1」を設定して、引数「map1」から 32bit の float 型の「X 座標値マップ」、引数「map2」から 32bit の float 型の「Y 座標値マップ」を取得してください。

・32bit の float 型の X 座標値マップ、Y 座標値マップを、本機能で使用できる X 座標値マップ、Y 座標値マップに変換する例

```
int dst_width = 640;
int dst_height = 480;
float map_fdata;
Mat src; // 入力画像
Mat mapx_32f, mapy_32f; // 32bit の float 型の X 座標値マップ、Y 座標値マップ
Mat mapx = Mat::zeros(Size(dst_width, dst_height), CV_16U); // 本機能で使用できる X 座標値マップ
Mat mapy = Mat::zeros(mapx.size(), CV_16U); // 本機能で使用できる Y 座標値マップ

for (int y = 0; y < mapx.rows; y++)
{
    for (int x = 0; x < mapx.cols; x++)
    {
        map_fdata = mapx_32f.at<float>(y, x);
        if ((map_fdata >= 0) && (map_fdata <= (src.cols - 1)))
        {
            mapx.at<unsigned short>(y, x) = (unsigned short)round((map_fdata * 32));
        }
        else
        {
            mapx.at<unsigned short>(y, x) = 0xffff;
        }

        map_fdata = mapy_32f.at<float>(y, x);
        if ((map_fdata >= 0) && (map_fdata <= (src.rows - 1)))
        {
            mapy.at<unsigned short>(y, x) = (unsigned short)round((map_fdata * 32));
        }
        else
        {
            mapy.at<unsigned short>(y, x) = 0xffff;
        }
    }
}
```

本機能は、OpenCV の cv::remap 関数の引数に以下を適用した場合と同等の結果が得られます。

・map1 に mapx の各データを 32 で割った後 float 型に変換したマップを適用
・map2 に mapy の各データを 32 で割った後 float 型に変換したマップを適用
・interpolation に INTER_LINEAR を指定
・borderMode に BORDER_CONSTANT を指定
・borderValue に border_value を指定

参考 URL：https://opencv.org/

注意	なし

4.10 Image filter

4.10.1 MedianBlur

MedianBlur

画像のノイズを除去します (Noise reduction)

コンフィグレーションデータファイル			r_drp_median_blur.dat	
対応バージョン			1.00	
コンフィグレーションデータサイズ（バイト）			57312	
ヘッダファイル			r_drp_median_blur.h	
パラメータ	構造体名			
	r_drp_gaussian_blur_t			
	メンバ名	型	説明	
	src	uint32_t	入力画像のアドレス	
	dst	uint32_t	出力画像のアドレス	
	width	uint16_t	画像の幅（ピクセル）	
	height	uint16_t	画像の高さ（ピクセル）	
	top	uint8_t	1:上端の境界処理あり。 0:上端の境界処理なし。 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。	
	bottom	uint8_t	1:下端の境界処理あり。 0:下端の境界処理なし。 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。	
入出力詳細	入力画像	アドレス	： src で指定	
		幅（ピクセル）	： width で指定（24～1280）	
		高さ（ピクセル）	： height で指定（8～960）	
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
	出力画像	アドレス	： dst で指定	
		幅（ピクセル）	： 入力画像と同じ	
		高さ（ピクセル）	： 入力画像と同じ	
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
タイル数	1			
分割処理	可			
解説	本機能は、src で指定したアドレスの画像を、メディアンフィルタを用いて平滑化を行い、dst で指定したアドレスに出力します。 メディアンフィルタは、画像または信号からノイズを除去するためによく使用される非線形デジタルフィルタの一つです。 本機能では、注目ピクセルを、周辺 9 ピクセルの中央値に置き換えます。周辺 9 ピクセルとは、注目ピクセルを中心とした 3 × 3 ピクセルです。 本機能は OpenCV の cv::medianBlur 関数の引数 ksize に 3 を指定した場合と同等の結果が得られます。 参考 URL： https://opencv.org/ 本機能は、分割処理を行わない場合、src と dst に同一アドレスを指定することが可能です。			
注意	なし			

4.10.2 GaussianBlur

GaussianBlur

画像を平滑化します (Smoothing)

コンフィグレーションデータファイル			r_drp_gaussian_blur.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			58432
ヘッダファイル			r_drp_gaussian_blur.h
パラメータ	構造体名		
	r_drp_gaussian_blur_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	top	uint8_t	1:上端の境界処理あり 0:上端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
	bottom	uint8_t	1:下端の境界処理あり 0:下端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
入出力詳細	入力画像	アドレス	: src で指定
		幅（ピクセル）	: width で指定（16～1280）
		高さ（ピクセル）	: height で指定（8～960）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
	出力画像	アドレス	: dst で指定
		幅（ピクセル）	: 入力画像と同じ
		高さ（ピクセル）	: 入力画像と同じ
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
タイル数	1		
分割処理	可		
解説	本機能は、src で指定したアドレスの画像にガウシアンフィルタを用いて平滑化を行い、dst で指定されたアドレスに出力します。 ガウシアンフィルタは、ガウス分布を利用した画像の平滑化に使われるフィルタの一つで、注目ピクセルに近いほど重みを大きくするカーネルを用いた画像フィルタです。本機能では、以下のようなカーネルを用います。 1/16　2/16　1/16 2/16　4/16　2/16 1/16　2/16　1/16 注目ピクセルを計算するために、注目ピクセルを中心とした 3 × 3 ピクセルをカーネルによる重みを付けて加算します。 本機能は、OpenCV の cv::GaussianBlur 関数の引数 ksize.width に 3、ksize.height に 3、sigmaX に 1.3、sigmaY に 1.3、borderType に BORDER_REFLECT_101 を指定した場合と同等の結果が得られます。 参考 URL： https://opencv.org/ 本機能は、分割処理を行わない場合、src と dst に同一アドレスを指定することが可能です。		
注意	なし		

4.10.3 UnsharpMasking

UnsharpMasking

画像を鮮鋭化します (Sharpening)

コンフィグレーションデータファイル		r_drp_unsharp_masking.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		156096	
ヘッダファイル		r_drp_unsharp_masking.h	
パラメータ	構造体名		
	r_drp_unsharp_masking_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	strength	uint8_t	フィルタの強調度の値（0～255） （詳細は解説を参照してください）
	top	uint8_t	1:上端の境界処理あり。 0:上端の境界処理なし。 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
	bottom	uint8_t	1:下端の境界処理あり。 0:下端の境界処理なし。 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
入出力詳細	入力画像	アドレス	: src で指定
		幅（ピクセル）	: width で指定（16～1280）
		高さ（ピクセル）	: height で指定（8～960）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
	出力画像	アドレス	: dst で指定
		幅（ピクセル）	: 入力画像と同じ
		高さ（ピクセル）	: 入力画像と同じ
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
タイル数	2		
分割処理	可		

解説	本機能は src で指定された画像に鮮鋭化（エッジ強調）を行い、dst で指定したアドレスに出力します。 strength パラメータによって、強調の度合いを調整することが可能です。strength の値が大きいほど画像のエッジが強調されます。 UnsharpMasking は、OpenCV では cv::filter2D 関数を使用して、以下の係数を用いるのが一般的です。 （k は鮮鋭化の強さを表す係数。0 だと鮮鋭化なし）

-k/9	-k/9	-k/9
-k/9	1+(8*k/9)	-k/9
-k/9	-k/9	-k/9

参考 URL： https://opencv.org/

本機能では、上記の係数を固定小数で近似した、以下の係数を用いています。
k'は strength として指定します。

-k'/256	-k'/256	-k'/256
-k'/256	(9*28+(8*k'))/256	-k'/'256
-k'/256	-k'/256	-k'/256

strength に、k の値を 28 倍した値を指定することで、UnsharpMasking が行えます。
例えば、strength に 28 を指定すると、k=1.0 で UnsharpMasking を行ったものと同等の結果となります。

本機能は、分割処理を行わない場合、src と dst に同一アドレスを指定することが可能です。

注意	なし

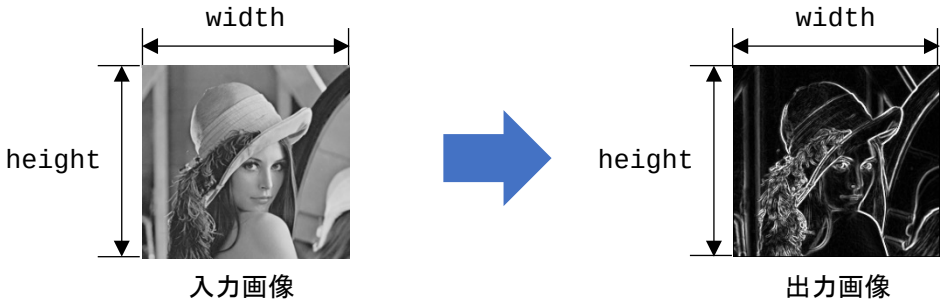
4.10.4 Sobel

Sobel

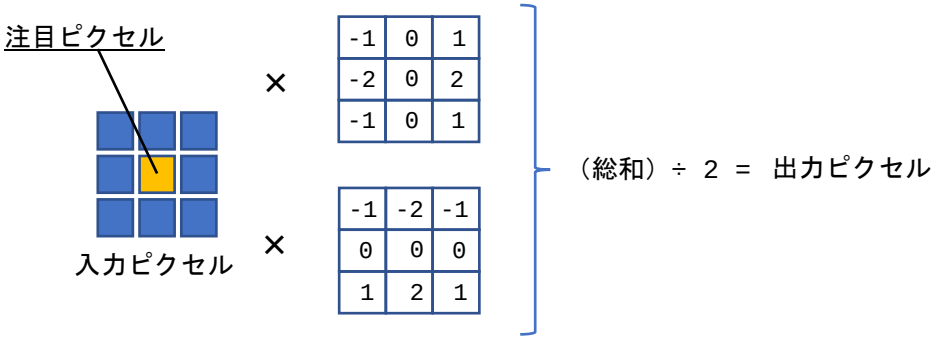
Sobel フィルタを使って輪郭を強調した画像を出力します

コンフィグレーションデータファイル			r_drp_sobel.dat	
対応バージョン			1.00	
コンフィグレーションデータサイズ（バイト）			40160	
ヘッダファイル			r_drp_sobel.h	
パラメータ	構造体名			
	r_drp_sobel_t			
	メンバ名	型	説明	
	src	uint32_t	入力画像のアドレス	
	dst	uint32_t	出力画像のアドレス	
	width	uint16_t	画像の幅（ピクセル）	
	height	uint16_t	画像の高さ（ピクセル）	
	top	uint8_t	1:上端の境界処理あり 0:上端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。	
	bottom	uint8_t	1:下端の境界処理あり 0:下端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。	
入出力詳細	入力画像	アドレス	： src で指定	
		幅（ピクセル）	： width で指定（16～1280）	
		高さ（ピクセル）	： height で指定（8～960）	
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
	出力画像	アドレス	： dst で指定	
		幅（ピクセル）	： 入力画像と同じ	
		高さ（ピクセル）	： 入力画像と同じ	
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
タイル数	1			
分割処理	可			

解説　本機能は、src で指定したアドレスの画像に Sobel フィルタを用いてエッジの強調を行い、dst で指定したアドレスに出力します。



本機能では、注目ピクセルの周囲1ピクセルの範囲（3x3ピクセルの範囲）に対して、水平／垂直方向のエッジを強調するために以下の計算を行います。



本機能は、分割処理を行わない場合、src と dst に同一アドレスを指定することが可能です。

注意　なし

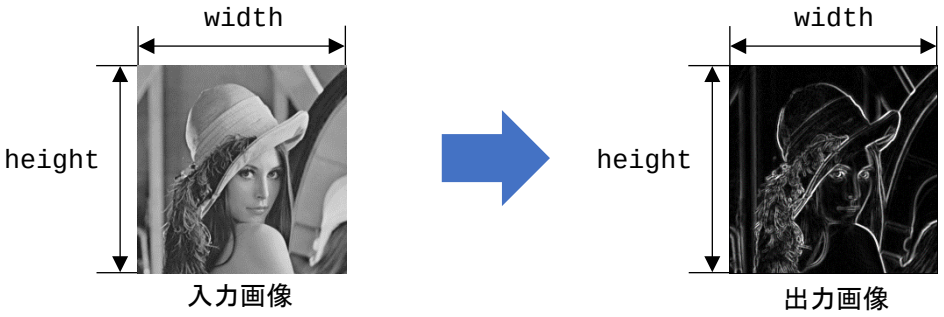
4.10.5 Prewitt

Prewitt

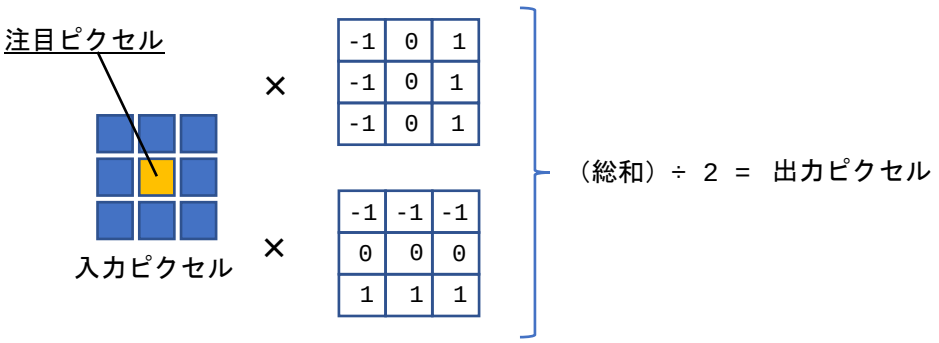
Prewitt フィルタを使って輪郭を強調した画像を出力します

コンフィグレーションデータファイル			r_drp_prewitt.dat	
対応バージョン			1.00	
コンフィグレーションデータサイズ（バイト）			40160	
ヘッダファイル			r_drp_prewitt.h	
パラメータ	構造体名			
	r_drp_prewitt_t			
	メンバ名	型	説明	
	src	uint32_t	入力画像のアドレス	
	dst	uint32_t	出力画像のアドレス	
	width	uint16_t	画像の幅（ピクセル）	
	height	uint16_t	画像の高さ（ピクセル）	
	top	uint8_t	1:上端の境界処理あり 0:上端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。	
	bottom	uint8_t	1:下端の境界処理あり 0:下端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。	
入出力詳細	入力画像	アドレス	： src で指定	
		幅（ピクセル）	： width で指定（16～1280）	
		高さ（ピクセル）	： height で指定（8～960）	
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
	出力画像	アドレス	： dst で指定	
		幅（ピクセル）	： 入力画像と同じ	
		高さ（ピクセル）	： 入力画像と同じ	
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
タイル数	1			
分割処理	可			

解説　本機能は、src で指定したアドレスの画像に Prewitt フィルタを用いてエッジの強調を行い、dst で指定したアドレスに出力します。



本機能では、注目ピクセルの周囲 1 ピクセルの範囲（3x3 ピクセルの範囲）に対して、水平／垂直方向のエッジを強調するために以下の計算を行います。



本機能は、分割処理を行わない場合、src と dst に同一アドレスを指定することが可能です。

注意　なし

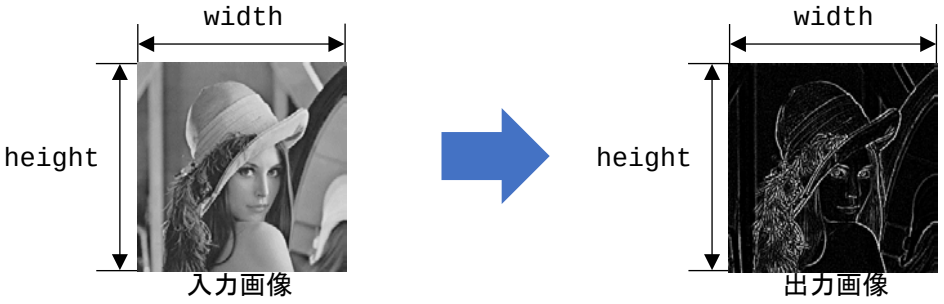
4.10.6 Laplacian

Laplacian

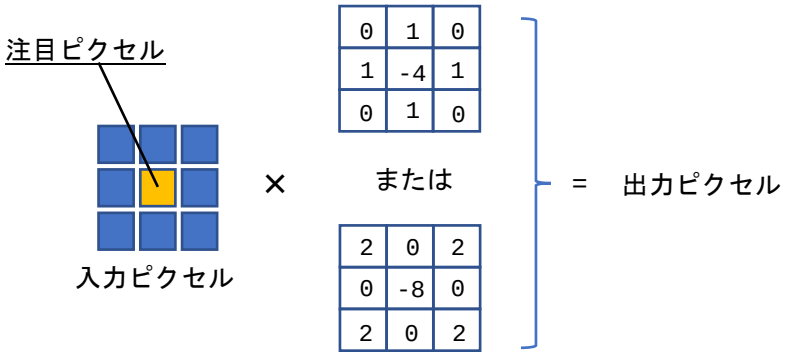
Laplacian フィルタを使って輪郭を強調した画像を出力します

コンフィグレーションデータファイル			r_drp_laplacian.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			36704
ヘッダファイル			r_drp_laplacian.h
パラメータ	構造体名		
	r_drp_laplacian_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	top	uint8_t	1:上端の境界処理あり 0:上端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
	bottom	uint8_t	1:下端の境界処理あり 0:下端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
	kernel	uint8_t	0:フィルタ係数に $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ を使用 1:フィルタ係数に $\begin{pmatrix} 2 & 0 & 2 \\ 0 & -8 & 0 \\ 2 & 0 & 2 \end{pmatrix}$ を使用 0 か 1 のどちらかを指定してください。
入出力詳細	入力画像	アドレス	: src で指定
		幅（ピクセル）	: width で指定（16～1280）
		高さ（ピクセル）	: height で指定（8～960）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
	出力画像	アドレス	: dst で指定
		幅（ピクセル）	: 入力画像と同じ
		高さ（ピクセル）	: 入力画像と同じ
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
タイル数	1		
分割処理	可		

解説　本機能は、src で指定したアドレスの画像に Laplacian フィルタを用いてエッジの強調を行い、dst で指定したアドレスに出力します。



本機能では、注目ピクセルの周囲 1 ピクセルの範囲（3x3 ピクセルの範囲）に対して、エッジを強調するために以下の計算を行います。



本機能は、OpenCV の cv::Laplacian 関数の引数 ddepth に CV_8U、ksize に 1 または 3、scale に 1.0、delta に 0、borderType に BORDER_REFLECT_101 を指定した場合と同等の結果が得られます。ksize=1 は、本機能では kernel=0、ksize=3 は、本機能では kernel=1 に該当します。

参考 URL： https://opencv.org/

本機能は、分割処理を行わない場合、src と dst に同一アドレスを指定することが可能です。

注意　なし

4.10.7 Dilate

Dilate

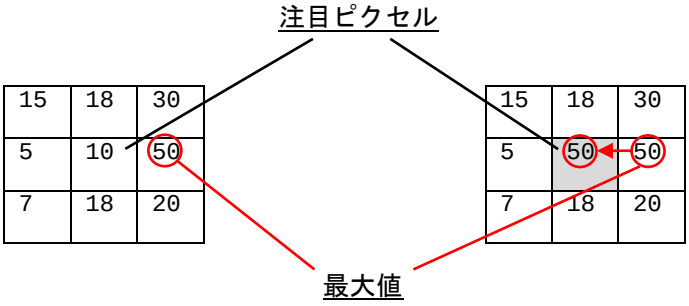
画像の白い部分を膨張させます

コンフィグレーションデータファイル		r_drp_dilate.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		56320	
ヘッダファイル		r_drp_dilate.h	
パラメータ	構造体名		
	r_drp_dilate_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	top	uint8_t	1:上端の境界処理あり 0:上端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
	bottom	uint8_t	1:下端の境界処理あり 0:下端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
入出力詳細	入力画像	アドレス	: src で指定
		幅（ピクセル）	: width で指定（16～1280）
		高さ（ピクセル）	: height で指定（8～960）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
	出力画像	アドレス	: dst で指定
		幅（ピクセル）	: 入力画像と同じ
		高さ（ピクセル）	: 入力画像と同じ
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
タイル数	1		
分割処理	可		

解説

本機能は、src で指定したアドレスの画像の白部分を膨張し、dst で指定したアドレスに出力します。

本機能は、注目ピクセルを中心とした 3 × 3 ピクセルのうち、最大値を注目ピクセルに設定します。白黒の二値画像を入力した場合、白い部分が 1 ピクセル分外側に膨張したように見えます。
OpenCV の cv::dilate 関数で、境界処理を BORDER_REPLICATE にした場合と同様の処理です。
参考 URL：https://opencv.org/



入力画像が二値化画像の場合の処理例を示します。

入力画像　出力画像

本機能は、分割処理を行わない場合、src と dst に同一アドレスを指定することが可能です。

注意　なし

4.10.8 Erode

Erode

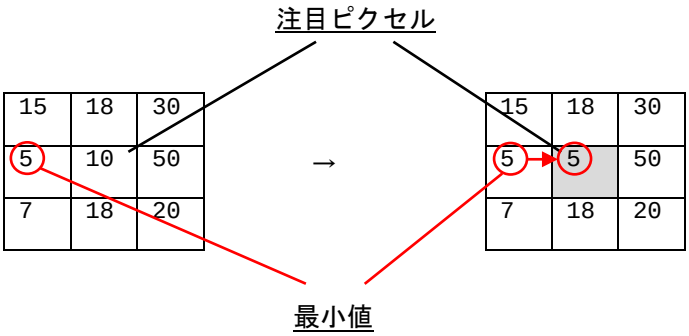
画像の白い部分を収縮させます

コンフィグレーションデータファイル			r_drp_erode.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			60352
ヘッダファイル			r_drp_erode.h
パラメータ	構造体名		
	r_drp_erode_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	top	uint8_t	1:上端の境界処理あり 0:上端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
	bottom	uint8_t	1:下端の境界処理あり。 0:下端の境界処理なし。 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
入出力詳細	入力画像	アドレス	: src で指定
		幅（ピクセル）	: width で指定（16～1280）
		高さ（ピクセル）	: height で指定（8～960）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
	出力画像	アドレス	: dst で指定
		幅（ピクセル）	: 入力画像と同じ
		高さ（ピクセル）	: 入力画像と同じ
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height) × 1 バイト
タイル数	1		
分割処理	可		

解説

本機能は、src で指定したアドレスの画像の白部分を収縮し、dst で指定したアドレスに出力します。

本機能は、注目ピクセルを中心とした 3 × 3 ピクセルのうち、最小値を注目ピクセルに設定します。白黒の二値画像を入力した場合、白い部分が 1 ピクセル分内側に収縮したように見えます。
OpenCV の cv::erode 関数で、境界処理を BORDER_REPLICATE にした場合と同様の処理です。
参考 URL： https://opencv.org/



入力画像が二値化画像の場合の処理例を示します。

本機能は、分割処理を行わない場合、src と dst に同一アドレスを指定することが可能です。

注意 なし

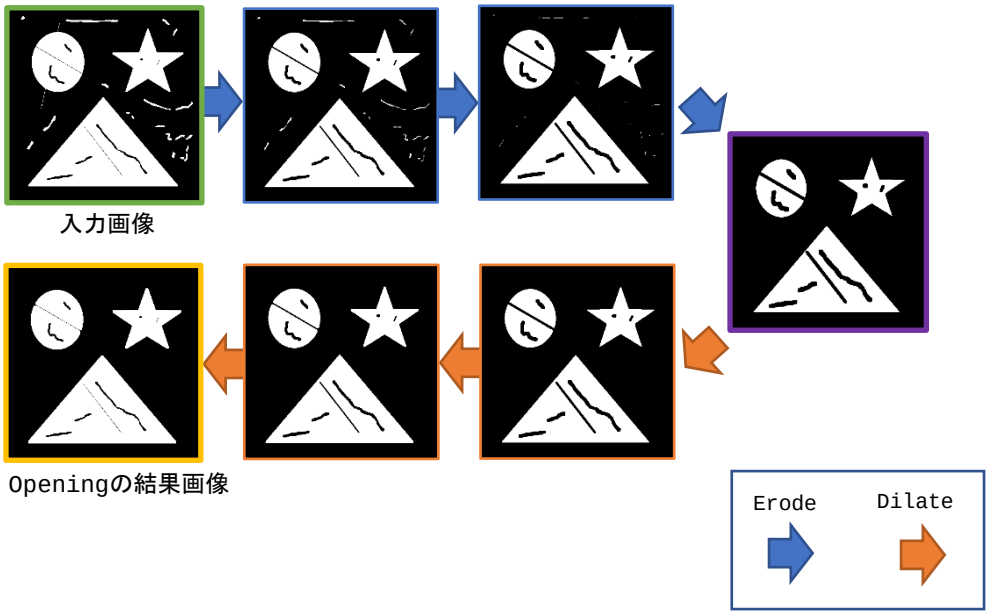
4.10.9 Opening

Opening

収縮(Erode)のあとに膨張(Dilate)して、黒部分のノイズを除去します

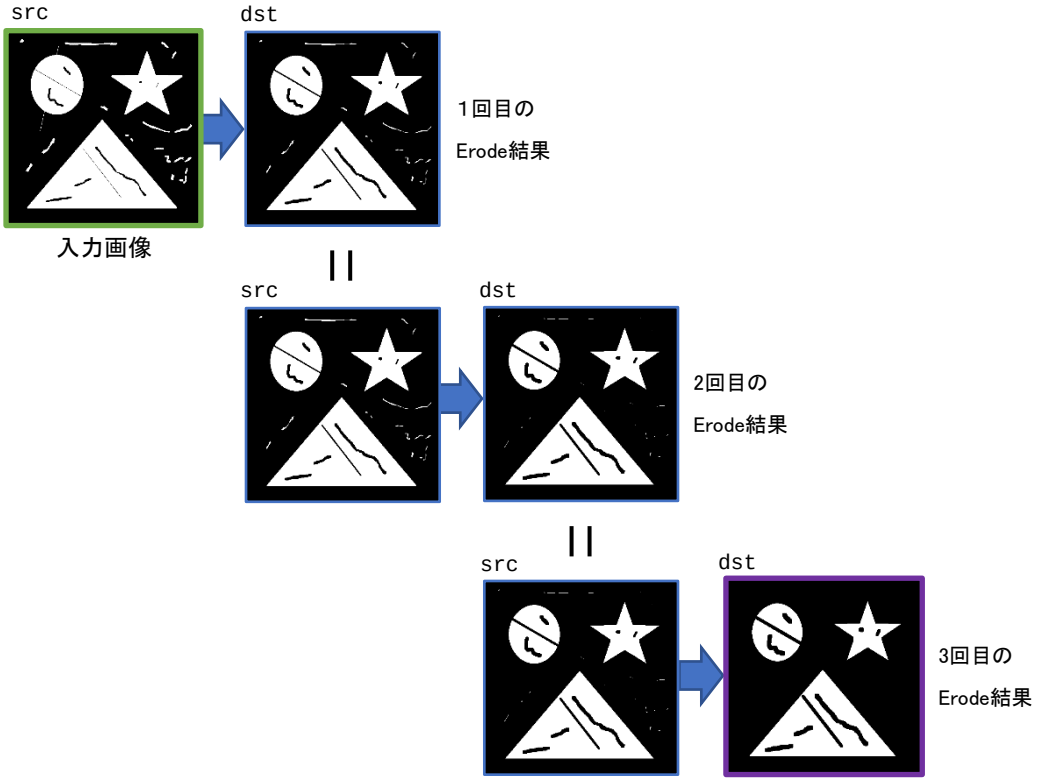
解説

本機能は、白部分の収縮を繰り返した後、膨張を繰り返す処理です。収縮と膨張は同じ回数を繰り返します。この機能はモノクロ画像のノイズ除去などに使用されます。
本機能は、DRP Library の Erode 機能と Dilate 機能の二つを組み合わせて実現する事が可能です。Erode 機能と Dilate 機能の仕様は、それぞれの章を参照してください。

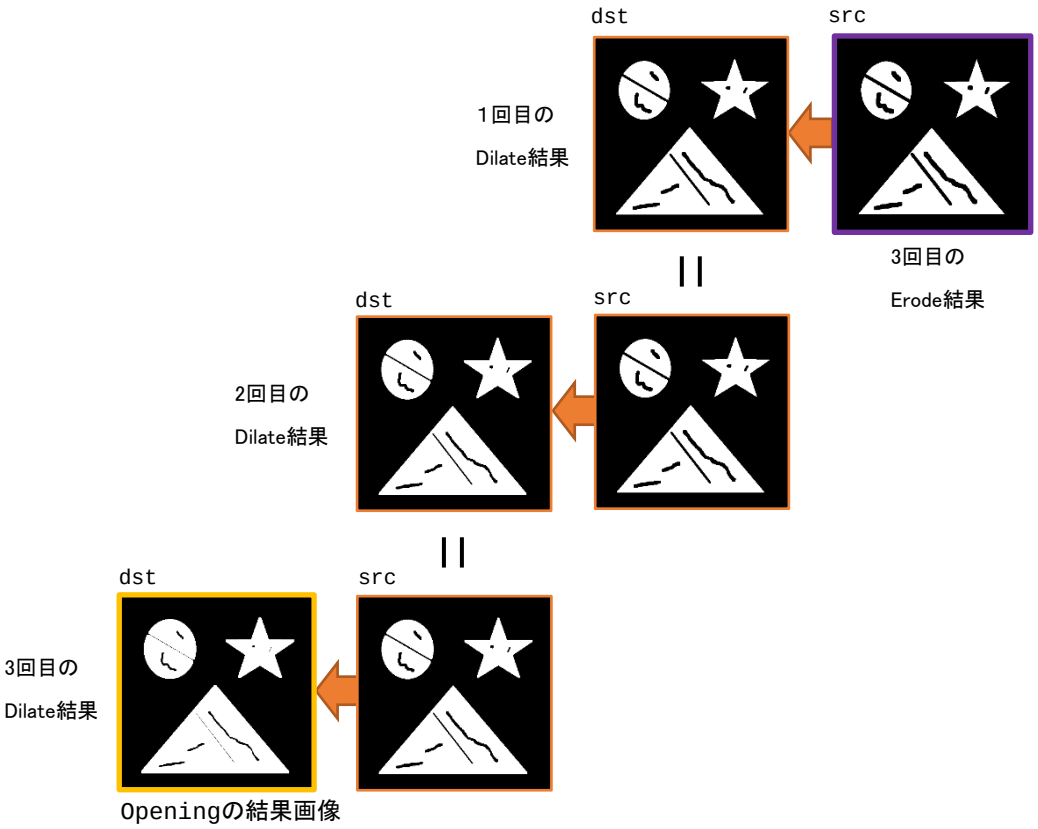


繰り返し回数が 3 の Opening 処理の方法を Erode と Dilate に分けて説明します。

【Erode】
Erode（収縮）を 3 回繰り返し処理するイメージを以下に示します。
1 回目の Erode では、入力する画像を入力画像に設定し Erode の処理をします。
2 回目の Erode では、1 回目の出力画像を入力画像に設定し Erode の処理をします。
3 回目の Erode では、2 回目の出力画像を入力画像に設定し Erode の処理をします。



【Dilate】
Dilate（膨張）を 3 回繰り返し処理するイメージを以下に示します。
1 回目の Dilate では、3 回目の Erode 結果を入力画像に設定し Dilate の処理をします。
2 回目の Dilate では、1 回目の出力画像を入力画像に設定し Dilate の処理をします。
3 回目の Dilate では、2 回目の出力画像を入力画像に設定し Dilate の処理をします。
3 回目の Dilate 結果が Opening の結果画像となります。



本機能は、OpenCV の cv::morphologyEx 関数の引数 op に MORPH_OPEN、 kernel に cv::Mat()、anchor に Point(-1,-1)、iterations に繰り返し回数、borderType に BORDER_REPLICATE を指定した場合と同等の結果が得られます。

参考URL： https://opencv.org/

注意	Erode や Dilate を分割処理で実行する場合は、分割された結果画像がすべて揃ってから、次の処理を行ってください。

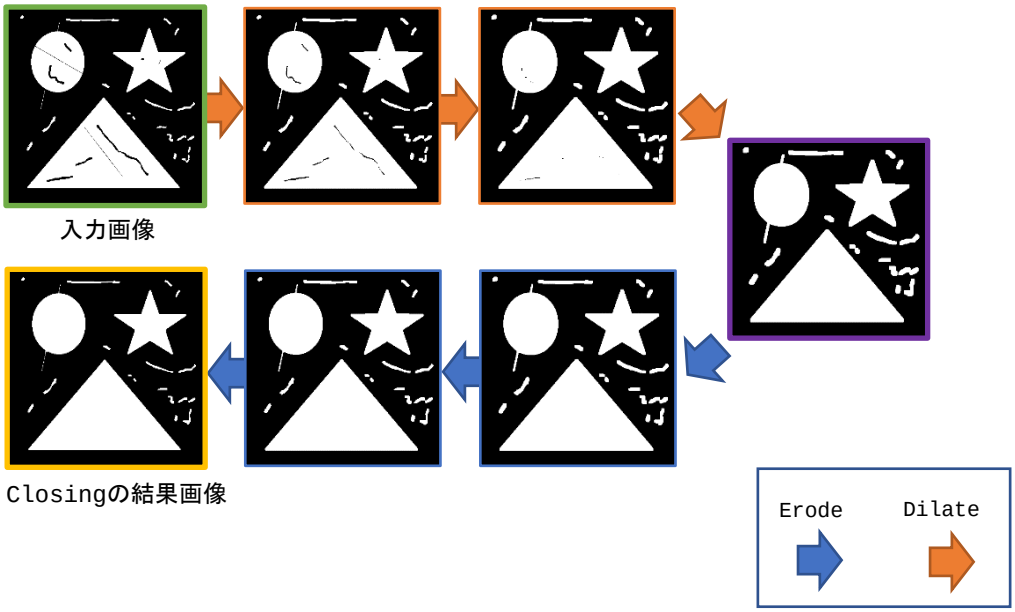
4.10.10 Closing

Closing

膨張(Dilate)のあとに収縮(Erode)して、白部分のノイズを除去します

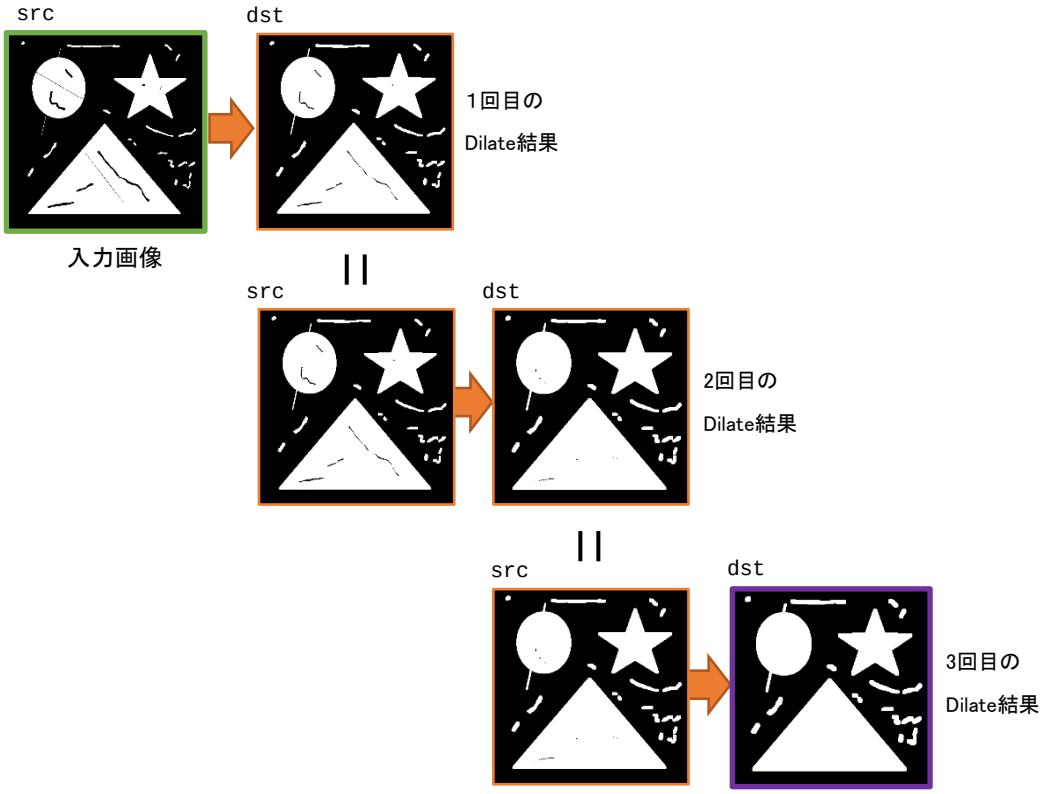
解説

本機能は、白部分の膨張を繰り返した後、収縮を繰り返す処理です。膨張と収縮は同じ回数を繰り返します。この機能はモノクロ画像のノイズ除去などに使用されます。
本機能は、DRP Library の Dilate 機能と Erode 機能の二つを組み合わせて実現する事が可能です。Dilate 機能と Erode 機能の仕様は、それぞれの章を参照してください。

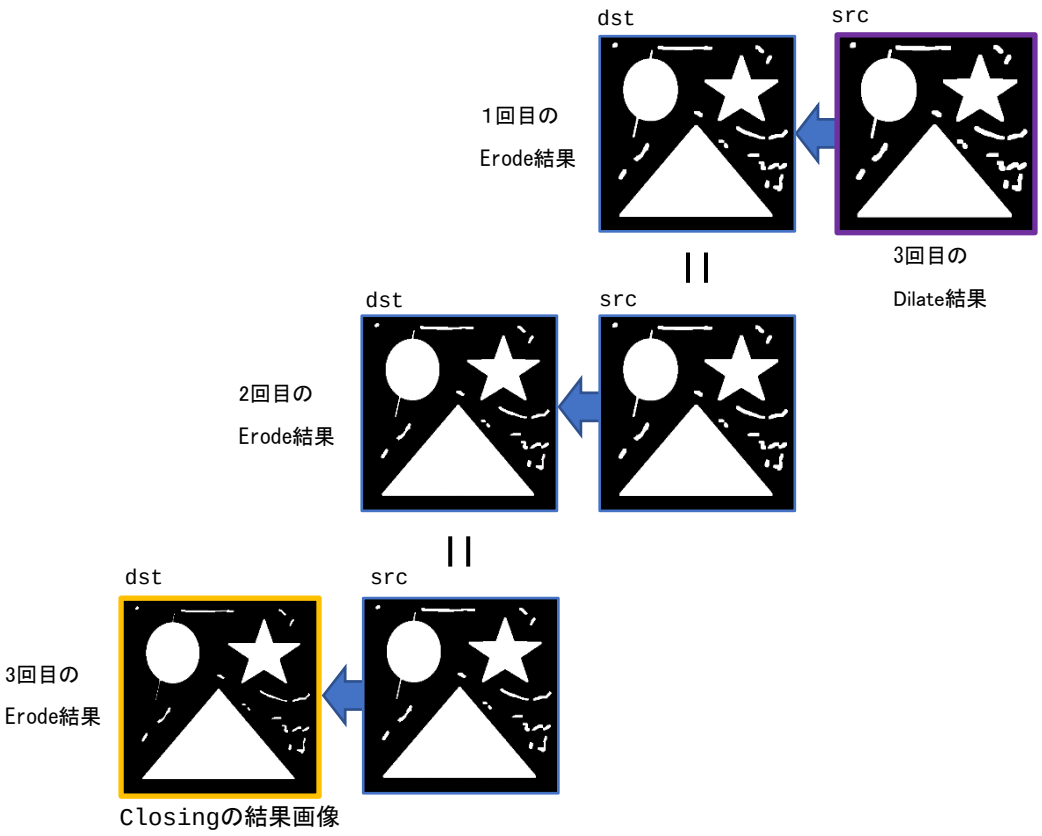


繰り返し回数が 3 の Closing 処理の方法を Dilate と Erode に分けて説明します。

【Dilate】
Dilate（膨張）を 3 回繰り返し処理するイメージを以下に示します。
1 回目の Dilate では、入力する画像を入力画像に設定し Dilate の処理をします。
2 回目の Dilate では、1 回目の出力画像を入力画像に設定し Dilate の処理をします。
3 回目の Dilate では、2 回目の出力画像を入力画像に設定し Dilate の処理をします。



【Erode】
Erode（収縮）を 3 回繰り返し処理するイメージを以下に示します。
1 回目の Erode では、3 回目の Dilate 結果を入力画像に設定し Erode の処理をします。
2 回目の Erode では、1 回目の出力画像を入力画像に設定し Erode の処理をします。
3 回目の Erode では、2 回目の出力画像を入力画像に設定し Erode の処理をします。
3 回目の Erode 結果が Closing の結果画像となります。



本機能は、OpenCV の cv::morphologyEx 関数の引数 op に MORPH_CLOSE、 kernel に cv::Mat()、anchor に Point(-1,-1)、iterations に繰り返し回数、borderType に BORDER_REPLICATE を指定した場合と同等の結果が得られます。

参考URL：https://opencv.org/

注意	Dilate や Erode を分割処理で実行する場合は、分割された結果画像がすべて揃ってから、次の処理を行ってください。

4.11 Feature detection

4.11.1 CannyCalculate

CannyCalculate

Canny 法によるエッジ判定

コンフィグレーションデータファイル		r_drp_canny_calculate.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		124736	
ヘッダファイル		r_drp_canny_calculate.h	
パラメータ	構造体名		
	r_drp_canny_calculate_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	work	uint32_t	ワークエリアのアドレス
	threshold_high	uint8_t	エッジ上限判定値（(threshold_low + 1)～255）
	threshold_low	uint8_t	エッジ下限判定値（0～(threshold_high - 1)）
	top	uint8_t	1:上端の境界処理あり 0:上端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
	bottom	uint8_t	1:下端の境界処理あり 0:下端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を、それ以外の場合は 0 を指定してください。
入出力詳細	入力画像	アドレス	： src で指定
		幅（ピクセル）	： width で指定（16～1280、16 の整数倍）
		高さ（ピクセル）	： height で指定（16～960、4 の整数倍）
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (width) × (height) × 1 バイト
	出力画像	アドレス	： dst で指定
		幅（ピクセル）	： 入力画像と同じ
		高さ（ピクセル）	： 入力画像と同じ
		フォーマット	： 8bit エッジ候補（0,1,2 の 3 種類） 0 ： 非エッジ 1 ： ウィークエッジ 2 ： ストロングエッジ （1 ピクセルあたり 1 バイト）
		データサイズ	： (width) × (height) × 1 バイト
	ワークエリア	アドレス	： work で指定
		データサイズ	： (((width) × (height + 2)) × 2)バイト
		＜内容＞ エッジの強さ、及び、エッジの方向を保存するための領域です。エッジの強さ、及び、エッジの方向については解説を参照してください。	
タイル数	2		
分割処理	可		

解説

本機能は、src で指定したアドレスの画像から Canny 法によりエッジ候補を求め、dst で指定されたアドレスに出力します。

Canny 法によるエッジ検出は、エッジの誤検出が少ない検出方法です。また、エッジを細い線として出力することが可能です。Canny 法では、以下の順で処理を行います。

1. ノイズ除去(ガウシアンフィルタ)を行う。
2. エッジ強度、方向を計算し、非極大値の抑制、エッジの分類を行う。
3. ヒステリシス閾値処理によりエッジを確定する。

OpenCV の cv::Canny()は上記 2,3 の処理を行います。本機能では 1 を GaussianBlur、2 を CannyCalculate、3 を CannyHysterisis で行うことにより、同様にエッジを出力することが可能です。

参考 URL： https://opencv.org/

本機能で出力したエッジ候補は、エッジの強さ(EDGE 強度)によってエッジなし、ウィークエッジ、ストロングエッジの 3 種類となります。パラメータの threshold_low、threshold_high でウィークエッジ、ストロングエッジと判断する閾値を設定します。閾値を小さくするほど、エッジ候補は多くなります。

入力画像

出力画像
threshold_low=0x18
threshold_high=0x30

出力画像
threshold_low=0x05
threshold_high=0x28

灰色：ウィークエッジ
白：ストロングエッジ
として表示した場合。

本機能では、以下のように EDGE 強度と方向を計算しています。

$$Gx = \begin{bmatrix} G_{00} & G_{01} & G_{02} \\ G_{10} & G_{11} & G_{12} \\ G_{20} & G_{21} & G_{22} \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$Gy = \begin{bmatrix} G_{00} & G_{01} & G_{02} \\ G_{10} & G_{11} & G_{12} \\ G_{20} & G_{21} & G_{22} \end{bmatrix} \times \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

EDGE 強度 = $((Gx)^2 + (Gy)^2) >> 7$

```
if ( 3 * abs(Gx) <= 8 * abs(Gy))      // 21 度以下
        EDGE 方向 = DIR0
else if (20 * abs(Gx) > 8 * abs(Gy)) // 67 度より大きい
        EDGE 方向 = DIR90
else
        EDGE 方向 = (sign(Gx)==sign(Gy)) ? DIR45 : DIR135
```

注意 なし

4.11.2 CannyHysterisis

CannyHysterisis

ヒステリシスによる閾値処理

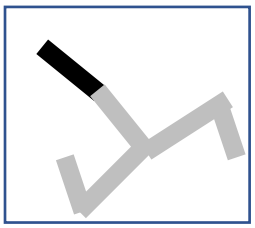
コンフィグレーションデータファイル			r_drp_canny_hysterisis.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			375776
ヘッダファイル			r_drp_canny_hysterisis.h
パラメータ	構造体名		
	r_drp_canny_hysterisis_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	work	uint32_t	ワークエリアのアドレス
	iterations	uint8_t	1～254 ： 繰り返し最大回数 255 ： 繰り返し回数無限
入出力詳細	入力画像	アドレス	： src で指定
		幅（ピクセル）	： width で指定（16～1280、8 の整数倍）
		高さ（ピクセル）	： height で指定（16～960、4 の整数倍）
		フォーマット	： エッジの候補（0,1,2 の 3 種類） 0 ： 非エッジ 1 ： ウィークエッジ 2 ： ストロングエッジ （1 ピクセルあたり 1 バイト）
		データサイズ	： (width) × (height) × 1 バイト
	出力画像	アドレス	： dst で指定
		幅（ピクセル）	： 入力画像と同じ
		高さ（ピクセル）	： 入力画像と同じ
		フォーマット	： 検出したエッジ（0,255 の 2 種類） 0 ： 非エッジ 255 ： エッジ （1 ピクセルあたり 1 バイト）
		データサイズ	： (width) × (height) × 1 バイト
	ワークエリア	アドレス	： work で指定
		データサイズ	： (width) × (height) × 1 バイト
		＜内容＞ ヒステリシス処理途中のデータを保存します。	
タイル数	6		
分割処理	不可		

解説

本機能は src で指定された画像(エッジ候補)に対して、ヒステリシス閾値処理を行い、dst で指定されたアドレスにエッジ画像を出力します。(Canny 法によるエッジ検出の後半処理部分です。詳細は、CannyCalculate の項を参照してください)

ヒステリシス閾値処理では、入力されたエッジの候補から、ストロングエッジに接続しているウィークエッジはエッジとして出力、ストロングエッジに接続していないウィークエッジは非エッジとして出力します。

下方向、上方向と交互にエッジの接続確認を行います。ウィークエッジがエッジとなった場合は、そのエッジに接続しているウィークエッジの確認が必要なため、サーチを最大 iterations 回まで繰り返します。ストロングエッジに変更されないサーチが 2 回続くと終了します。（処理時間と精度によって値を設定してください。）

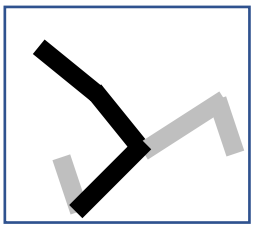


入力画像

灰色：ウィークエッジ

黒：ストロングエッジ

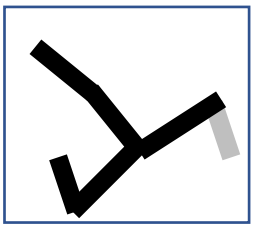
として表示した場合。



サーチ1回目

(下方向にサーチ)

ウィークエッジをストロングエッジに変更したので継続



サーチ2回目

(上方向にサーチ)

ウィークエッジをストロングエッジに変更したので継続

・・・

最大iterations回

入力画像

灰色：ウィークエッジ

白：ストロングエッジ

として表示した場合。

出力画像

注意 なし

4.11.3 CornerHarris

CornerHarris

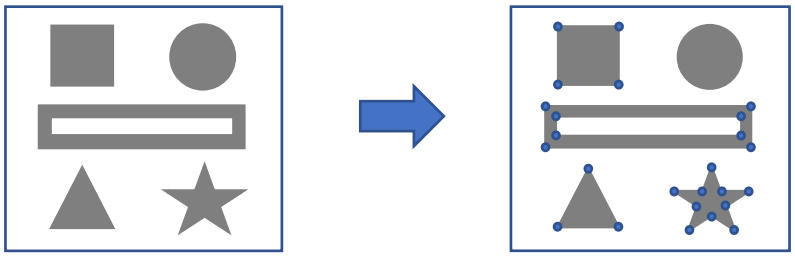
Chris Harris の考案した手法で画像に含まれる頂点を検出します

コンフィグレーションデータファイル			r_drp_corner_harris.dat	
対応バージョン			1.00	
コンフィグレーションデータサイズ（バイト）			408064	
ヘッダファイル			r_drp_corner_harris.h	
パラメータ	構造体名			
	r_drp_corner_harris_t			
	メンバ名	型	説明	
	src	uint32_t	入力画像のアドレス	
	dst	uint32_t	出力画像のアドレス Harris 検出器の応答が格納される。	
	width	uint16_t	画像の幅（ピクセル）	
	height	uint16_t	画像の高さ（ピクセル）	
	shift	uint8_t	Harris 検出器の応答の右シフト量 本機能は 32 ビットの Harris 検出器の応答を、本引数で指定された量右シフトして、0～255 に飽和演算して出力します。 Harris 検出器の応答は 256～65535 の間に収まることが多いため、8 を推奨値としています。	
入出力詳細	入力画像	アドレス	： src で指定	
		幅（ピクセル）	： width で指定（16～1280）	
		高さ（ピクセル）	： height で指定（8～960）	
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
	出力画像（Harris 検出器の応答）	アドレス	： dst で指定	
		幅（ピクセル）	： 入力画像と同じ	
		高さ（ピクセル）	： 入力画像と同じ	
		フォーマット	： 頂点検出結果（0～255） 値が大きいほど頂点である可能性が高いことを表します。 （1 ピクセルあたり 1 バイト）	
		データサイズ	： (width) × (height) × 1 バイト	
タイル数	6			
分割処理	不可			

解説

本機能は、src で指定されたアドレスの画像に対して、Harris 検出器を適用して、画像内の頂点を検出し、結果を dst で指定されたアドレスに出力します。

Harris 検出器では、注目ピクセル近辺の特徴が周辺の特徴と異なっていることを認識して、頂点を認識します。



入力画像から頂点を検出した模式図

Harris 検出器の計算方法は、3 × 3 ピクセルの近傍領域全体にわたって勾配の積和を計算することで注目ピクセルにおける、2 × 2 の勾配分散行列 $M^{(x,y)}$ を求めます。そこから次の特徴量を計算します。

$$\mathrm{dst}(x,y) = \det M^{(x,y)} - k\left(\mathrm{tr}M^{(x,y)}\right)^2$$

k はコーナー検出量固有係数で経験的に 0.04～0.15 が良いとされています。本機能では、0.0625 としています。

本機能は OpenCV の cv::cornerHarris 関数の引数 blockSize に 3、apertureSize に 3、k に 0.0625、borderType に BORDER_REFLECT_101 を指定した場合と同等の結果が得られます。
参考 URL： https://opencv.org/

本機能は、src と dst に同一アドレスを指定することが可能です。

注意 なし

4.11.4 MinutiaeExtract

MinutiaeExtract

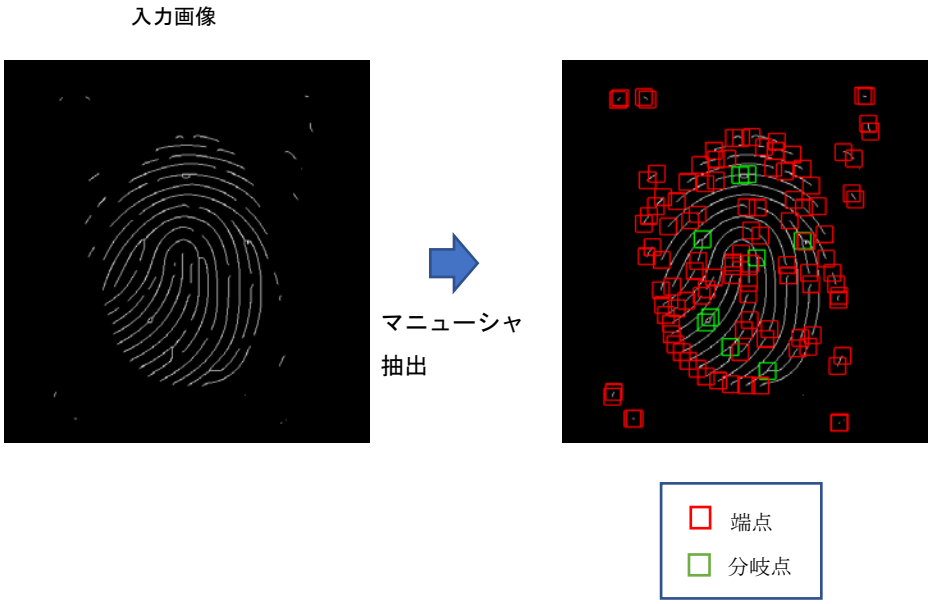
指紋認識で使用される指紋隆線の特徴点を抽出します

コンフィグレーションデータファイル		r_drp_minutiae_extract.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		132768	
ヘッダファイル		r_drp_minutiae_extract.h	
パラメータ	構造体名		
	r_drp_minutiae_extract_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	threshold	uint8_t	二値化の閾値（0～255）
	minutiae_data	uint32_t	特徴点データのアドレス
	minutiae_num	uint32_t	特徴点数のアドレス
	minutiae_max	uint32_t	特徴点の最大個数（1～2048）
	e_area_startx	uint16_t	抽出領域の開始位置の X 座標
	e_area_starty	uint16_t	抽出領域の開始位置の Y 座標
	e_area_width	uint16_t	抽出領域の幅
	e_area_height	uint16_t	抽出領域の高さ
入出力詳細	入力画像	アドレス : src で指定 幅（ピクセル） : width で指定（40～1280、4 の整数倍） 高さ（ピクセル） : height で指定（18～960） フォーマット : 8bit グレイスケール （1 ピクセルあたり 1 バイト） threshold 以下は黒、それ以外は白として扱います。 データサイズ : (width) × (height) × 1 バイト	
	抽出領域 （入力）	開始位置の X 座標 : e_area_startx で指定 （4 ～ width – 36、4 の倍数） 開始位置の Y 座標 : e_area_starty で指定 （1 ～ height – 17 ） 幅（ピクセル） : e_area_width で指定 （32 ～ width - e_area_startx - 4、4 の倍数） 高さ（ピクセル） : e_area_height で指定 （16 ～ height - e_area_starty - 1 ） ＜内容＞ 入力画像のマニューシャを抽出する領域です。 入力画像の左右の 4 ピクセルは、抽出領域として指定出来ません。 入力画像の上下の 1 ピクセルは、抽出領域として指定出来ません。 詳細は解説を参照してください。	

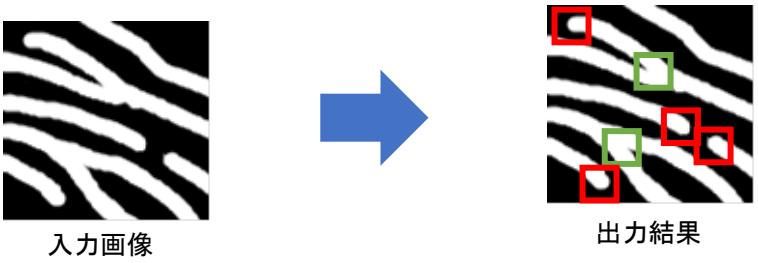
	特徴点データ （出力）	アドレス　：minutiae_data で指定 データサイズ　：minutiae_max ×8 バイト （32 ～ e_area_width × e_area_height × 8 ） フォーマット　：アドレス先頭から、 抽出した特徴点の X 座標値（2 バイト）、 抽出した特徴点の Y 座標値（2 バイト）、 抽出した特徴点の種別（1 バイト）、 抽出した特徴点の方向（1 バイト）、 0 パディング（2 バイト） ＜内容＞ アドレスの先頭から、特徴点の X 座標、Y 座標、種別、方向が 8 バイト単位で出力されます。 ただし、抽出した特徴点の数が minutiae_max を超えた場合は、minutiae_max 分の特徴点のみ出力します。 特徴点の種別は 0 が端点、1 が分岐となります。 特徴点の方向は注目ピクセルの左上を 0 として、時計回りに数値を振った時に白がある位置をビットで表現した 8 ビットの情報です。 詳細は解説を参照してください。
	特徴点数 （出力）	アドレス　：minutiae_num で指定 データサイズ　：4 バイト フォーマット　：抽出した特徴点の数（4 バイト） ＜内容＞ 抽出した特徴点の数が出力されます。 抽出した特徴点の数が minutiae_max を超えた場合でも、実際に抽出した特徴点数を出力します。 詳細は解説を参照してください。
タイル数	3	
分割処理	不可 ただし CPU 処理と組み合わせる事で分割処理が可能です。 詳細は解説を参照してください。	

解説

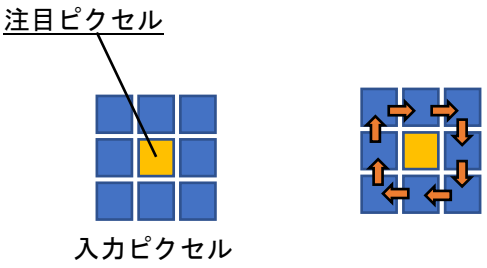
本機能は、src で指定したアドレスの入力画像を二値化して、マニューシャ抽出により画像の端点と分岐点を抽出します。正しいマニューシャ抽出を行うために、入力画像は細線化されている必要があります。抽出結果は minutiae_data と minutiae_num に特徴点の情報と個数を出力します。
本機能はマニューシャ特徴点の抽出までを行うものです。抽出した特徴点は、一般には不要な特徴点を削除するなどの処理が必要になります。本ライブラリでは、MinutiaeDelete 機能でその一例を実現しています。詳細は MinutiaeDelete 機能の章を参照してください。



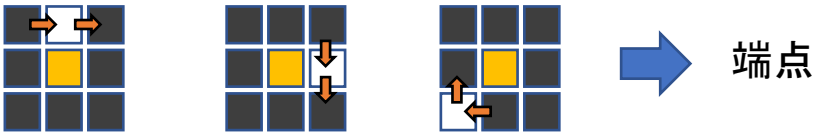
マニューシャ抽出では、細線化された画像の注目ピクセルと周囲のピクセルの情報に従って、指紋の特徴点を抽出します。特徴点とは、指紋の X 座標、Y 座標、端点か分岐点のどちらであるかの情報、端点と分岐点の方向です。



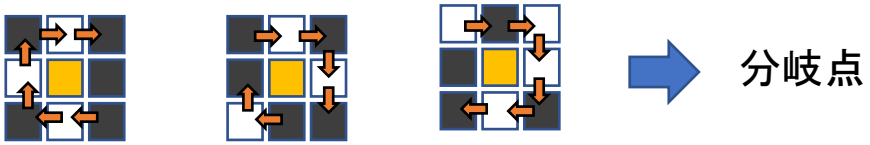
マニューシャ抽出では、注目ピクセルの周囲１ピクセルの範囲（3×3 ピクセルの範囲）に対して、隣にあるピクセルとの色の変化（白から黒、黒から白）をカウントし、2 回の場合は端点、5 回以上の場合は分岐点と判別します。
白と黒の判定は、threshold によって行われ、入力画像の輝度が threshold より大きければ白とします。



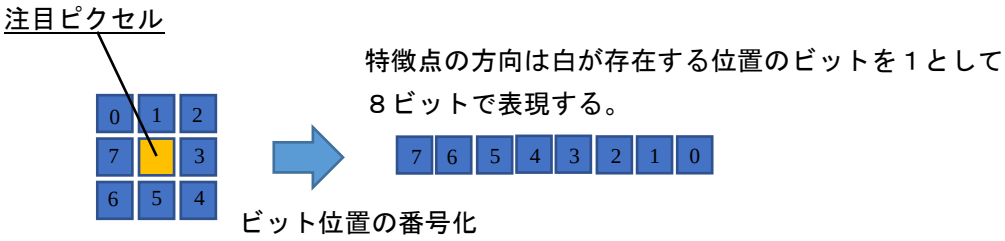
・変化が2回の場合例



・変化が5回以上の場合の例



マニューシャ抽出では、分岐と端点以外にも方向の情報を抽出します。方向は、マニューシャ抽出で注目ピクセルの周りの白の位置を 8 ビットの情報に変換して表現されます。



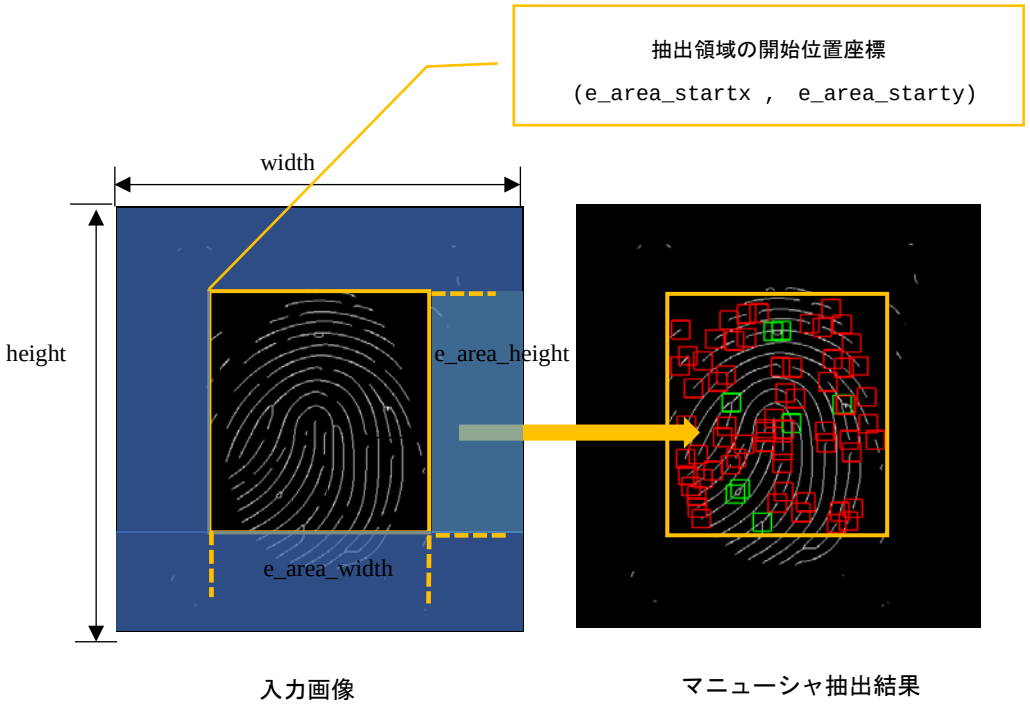
・端点と方向の例

方向＝0x02　方向＝0x08　方向＝0x40

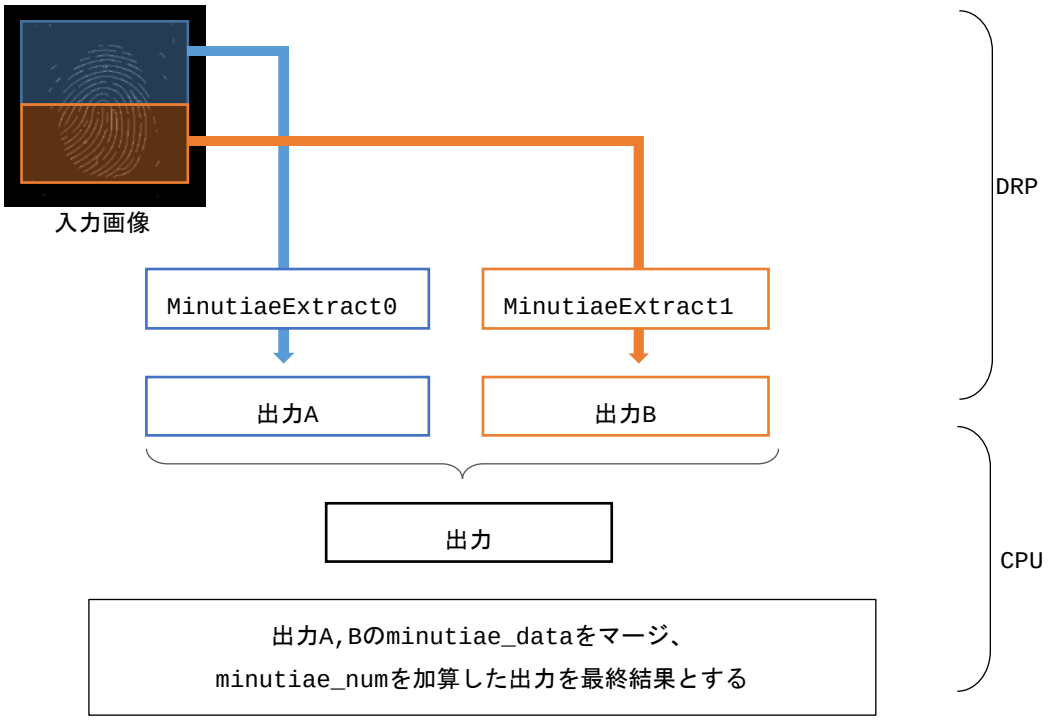
・分岐と方向の例

方向＝0xA2　方向＝0x25　方向＝0x2D

マニューシャ抽出では、src で指定したアドレスの画像に対し、実際にマニューシャ抽出を行う抽出領域を指定することができます。
これは、抽出領域の幅 e_area_width、高さ e_area_height、開始位置の座標 (e_area_startx, e_area_starty) によって領域を指定することができます。



本機能は、CPU による分割処理を行う事が出来ます。
2 並列処理を行う例を以下に示します。
分割前と同じマニューシャ抽出を実施するように、探索領域を 2 つの領域に分割し、MinutiaeExtract0、MinutiaeExtract1 に対して、それぞれ所定の minutiae_data、minutiae_num、minutiae_max、e_area_startx、e_area_starty、e_area_width、e_area_height を指定します。src、width、height、threshold は同じ設定としてください。
DRP によるマニューシャ抽出処理が完了した後、MinutiaeExtract0、MinutiaeExtract1 の minutiae_data 領域に出力された特徴点データをマージし、minutiae_num の特徴点数を加算する事で、分割処理を実現する事が可能です。



注意 なし

4.11.5 MinutiaeDelete

MinutiaeDelete

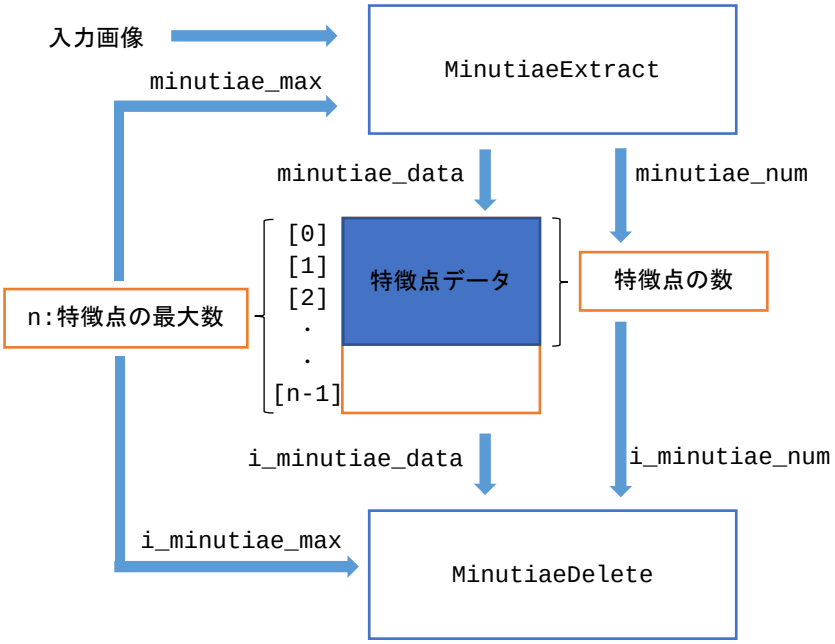
指紋認識で使用される指紋隆線の特徴点を削除します

コンフィグレーションデータファイル		r_drp_minutiae_delete.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		112480	
ヘッダファイル		r_drp_minutiae_delete.h	
パラメータ	構造体名		
	r_drp_minutiae_delete_t		
	メンバ名	型	説明
	trust_map	uint32_t	信用度情報のアドレス
	width	uint16_t	信用度情報の幅（ピクセル）
	height	uint16_t	信用度情報の高さ（ピクセル）
	i_minutiae_data	uint32_t	入力する特徴点データのアドレス
	i_minutiae_num	uint32_t	入力する特徴点数のアドレス
	i_minutiae_max	uint32_t	入力する特徴点数の最大数（1～2048）
	o_minutiae_data	uint32_t	出力される特徴点データのアドレス
	o_minutiae_num	uint32_t	出力される特徴点数のアドレス
	work	uint32_t	ワークエリアのアドレス
	第一除去		
	del1_distance	uint16_t	第一除去の距離指定（0～65535）
	del1_probability	uint8_t	第一除去の信用度情報指定（0～255）
	del1_bifurcation	uint8_t	第一除去の分岐の除去抑制指定（0～255）
	第二除去		
	del2_distance	uint16_t	第二除去の距離指定（0～65535）
	del2_count	uint8_t	第二除去の特徴点数指定（0～255）
	del2_bifurcation	uint8_t	第二除去の分岐の除去抑制指定（0～255）
	第三除去		
	del3_distance_s	uint16_t	第三除去の距離指定（同じ種別）（0～65535）
	del3_distance_d	uint16_t	第三除去の距離指定（違う種別）（0～65535）
	del3_probability	uint8_t	第三除去の信用度情報指定(0～255)
	del3_bifurcation	uint8_t	第三除去の分岐の除去抑制指定(0～255)
入出力詳細	入力特徴点数	アドレス : i_minutiae_num で指定 データサイズ : 4 バイト フォーマット : 削除処理する特徴点の数（4 バイト） ＜内容＞ 削除処理する特徴点の数を入力します。 ただし、i_minutiae_num で指定した値が i_minutiae_max を超えている場合、i_minutiae_max 分の特徴点のみ処理します。	

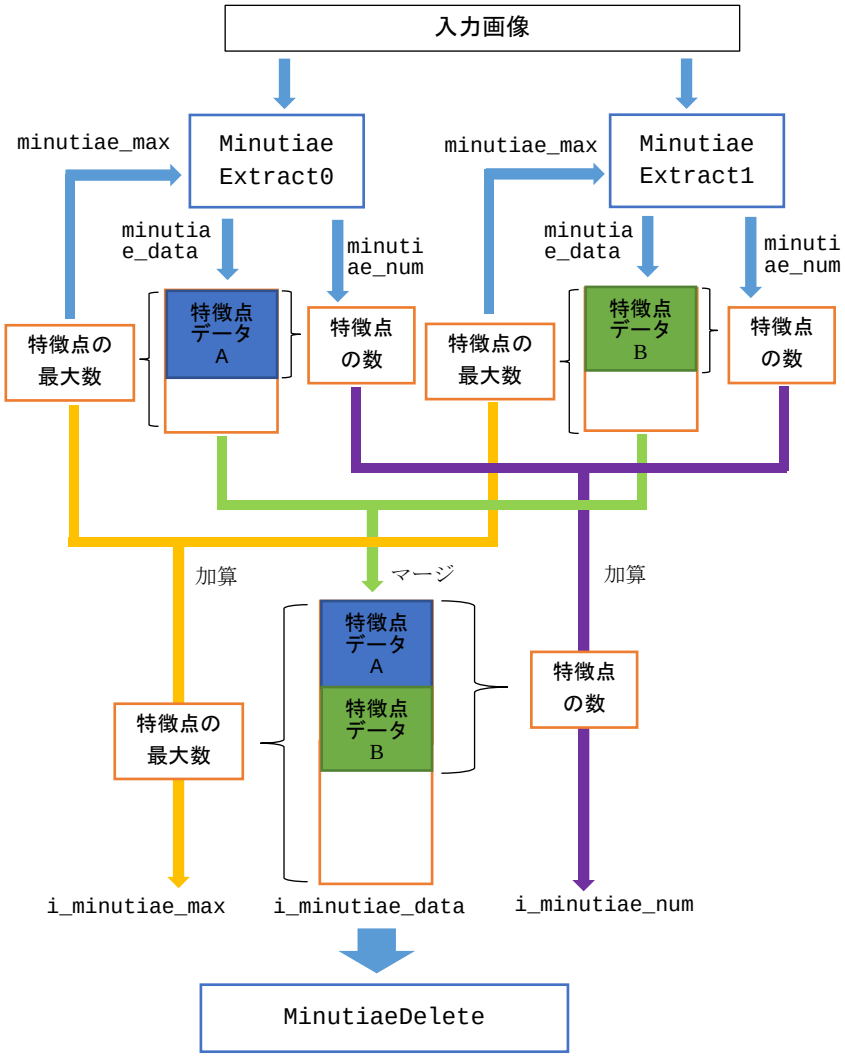
入力特徴点データ	アドレス	：	i_minutiae_data で指定
	データサイズ	：	入力特徴点数　×　8 バイト
	フォーマット	：	アドレス先頭から、 抽出した特徴点の X 座標値（2 バイト）、 抽出した特徴点の Y 座標値（2 バイト）、 抽出した特徴点の種別（1 バイト）、 抽出した特徴点の方向（1 バイト）、 0 パディング（2 バイト）

＜内容＞
アドレスの先頭から、特徴点の X 座標、Y 座標、種別、方向を 8 バイト単位で入力特徴点数だけ入力します。
MinutiaeExtract の結果を入力する場合は、i_minutiae_data に minutiae_data、i_minutiae_num に minutiae_num のアドレスを指定し、i_minutiae_max に miutiae_max の値を設定してください。

【分割処理を行わない場合】



【分割処理を行う場合】



MinutiaeExtract を分割処理した場合は、特徴点データをマージした結果を i_minutiae_data と i_minutiae_num で指定したアドレスに格納して下さい。
特徴点の種別は 0 が端点、1 が分岐となります。
特徴点の方向は注目ピクセルの左上を 0 として、時計回りに数値を振った時に白がある位置をビットで表現した 8 ビットの情報です。
詳細は MinutiaeExtract の解説を参照してください。

信用度情報 （入力）	アドレス 幅（ピクセル） 高さ（ピクセル） フォーマット データサイズ	： trust_map で指定 ： width で指定（40～1280、4 の整数倍） ： height で指定（18～960） ： 8bit （1 ピクセルあたり 1 バイト） ： (width) × (height) × 1 バイト

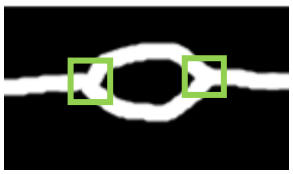
＜内容＞
特徴点が信用できるものであるかを判定するための情報です。
高い数値の場合、信用度が高いと判定されます。
詳細は解説を参照してください。

	出力特徴点数	アドレス : o_minutiae_num で指定 データサイズ : 4 バイト フォーマット : 削除処理後の特徴点の数（4 バイト） ＜内容＞ 削除処理後の特徴点の数が出力されます。
	出力特徴点データ	アドレス : o_minutiae_data で指定 データサイズ : i_minutiae_max×8 バイト フォーマット : アドレス先頭から、 抽出した特徴点の X 座標値（2 バイト）、 抽出した特徴点の Y 座標値（2 バイト）、 抽出した特徴点の種別（1 バイト）、 抽出した特徴点の方向（1 バイト）、 0 パディング（2 バイト） ＜内容＞ アドレスの先頭から、削除処理後の特徴点の X 座標、Y 座標、種別、方向が 8 バイト単位で出力特徴点数だけ出力されます。 出力特徴点数は、最大で i_minutiae_max と同じとなるため、データサイズは i_minutiae_max だけ持つ必要があります。 特徴点の種別は 0 が端点、1 が分岐となります。 特徴点の方向は注目ピクセルの左上を 0 として、時計回りに数値を振った時に白がある位置をビットで表現した 8 ビットの情報です。 詳細は MinutiaeExtract の解説を参照してください。
	ワークエリア	アドレス : work で指定 データサイズ : i_minutiae_max ×16 バイト ＜内容＞ マニューシャ削除の処理途中のデータを保存します。
タイル数	2	
分割処理	不可	

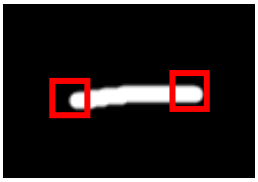
解説

本機能は、MinutiaeExtract で抽出した特徴点（i_minutiae_data と i_minutiae_num で指定）と信用度の情報を元に、第一除去、第二除去、第三除去を行い、o_minutiae_data と o_minutiae_num に特徴点の情報と個数を出力します。
特徴点数が 8 個以下になった場合は、除去処理を行いません。
第一除去、第二除去、第三除去までの工程をそれぞれに分けて解説します。

【第一除去】
第一除去では、以下２点のどちらかの条件を満たした特徴点を除去します。
・注目される特徴点から見て、同じ種類の特徴点が指定された距離に存在する
・信用度が低い
同じ種類の特徴点がある場合、線に穴が開いた状態か、短い線で真の特徴点ではない可能性があるためです。



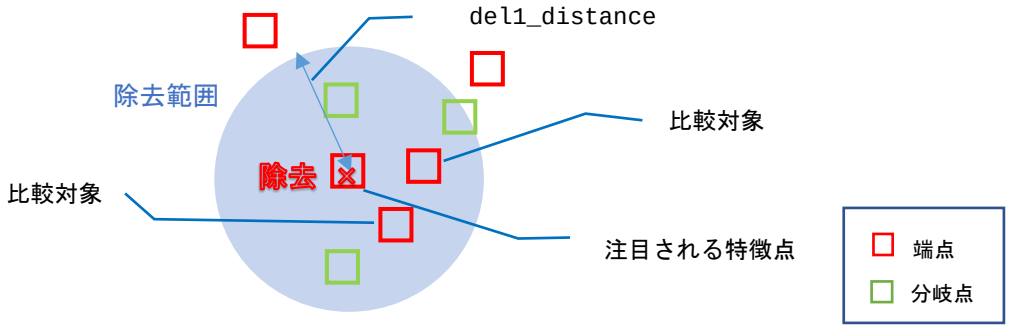
線に穴が開いた状態



短い線

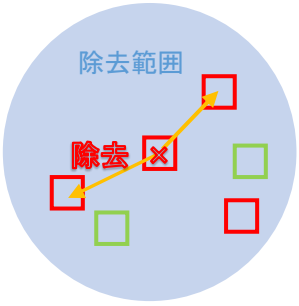
第一除去では、注目される特徴点（座標 XA、座標 YA）に対して、他の全ての特徴点（座標 XB、座標 YB）と比較して、特徴点の種類が同じで、del1_distance との関係が以下の条件を満たした時に、その特徴点を除去すると判定します。

$$(XA - XB)^2 + (YA - YB)^2 < del1\_distance$$

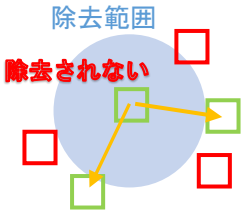


第一除去では、除去処理において分岐点の削除を抑制することが出来ます。
距離における除去を行う際は、分岐点の距離を del1_bifurcation 倍した値で除去判定を行います。
従って、分岐点における除去の条件は以下になります。

$$\{(XA - XB)^2 + (YA - YB)^2 \} \times del1\_bifurcation < del1\_distance$$

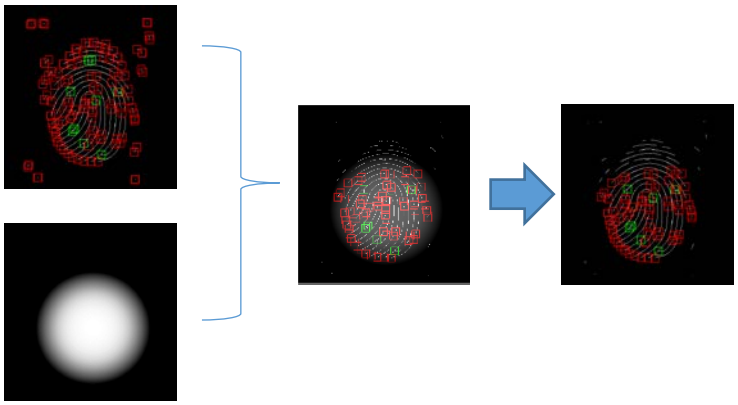


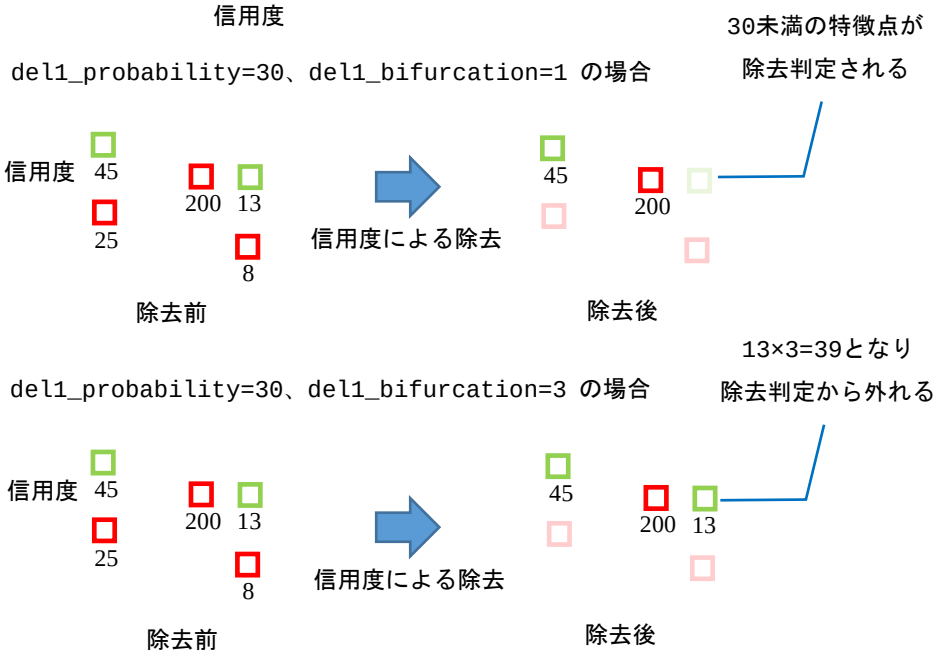
中心ピクセルが端点の場合



中心ピクセルが分岐点の場合
半径が1/[del1_bifurcation]倍されて
分岐点は削除されない

信用度が低い場合の除去では、注目される特徴点の信用度の情報が del1_probability より小さい場合、その特徴点は信用度の低い位置にいるため、除去されます。
なお、信用度による除去においても分岐点の除去は抑制されます。分岐点は、信用度の情報×del1_bifurcation が del1_probability より小さい場合に除去されます。





距離による除去を行わない場合は、del1_distance に 0 を指定します。信用度による除去を行わない場合は、 del1_probability に 0 を指定します。
del1_distance に 0 を指定し del1_probability に 0 を指定した場合は、第一除去の処理は行いません。

【第二除去】
第二除去では、以下の条件を満たした特徴点を除去します。
・注目される特徴点から見て、特徴点が指定された距離に指定された個数以上存在する

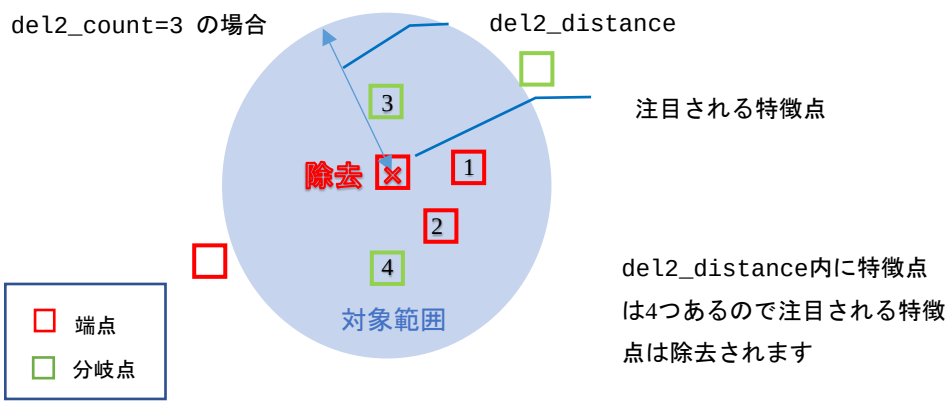
特徴点が集中するような場合、ノイズなどによって特徴点が発生している可能性があるためです。

第二除去では、注目される特徴点（座標 XA、座標 YA）に対して、他の全ての特徴点（座標 XB、座標 YB）と比較して、注目される特徴点が端点の場合は、以下の式を満たす特徴点が del2_count 個以上あった場合、注目される特徴点は除去されます。

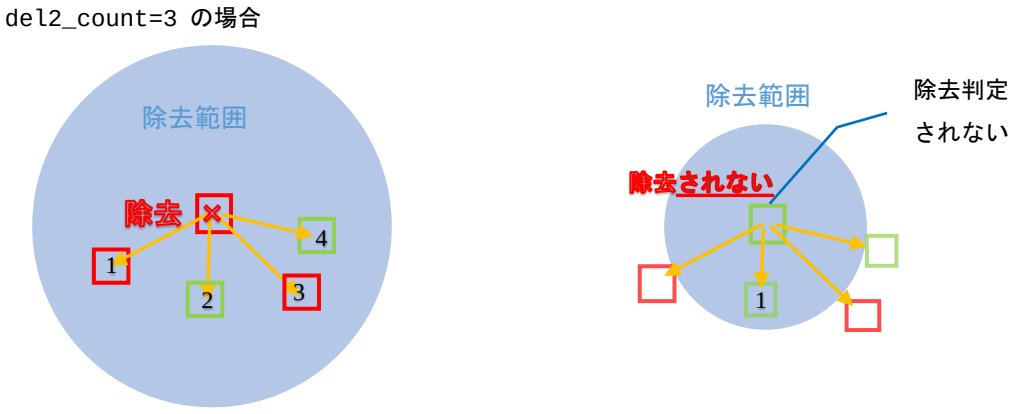
$$(XA - XB)^2 + (YA - YB)^2 < del2\_distance$$

注目される特徴点が分岐点の場合は、以下の式を満たす特徴点が del2_count 個以上あった場合、注目される特徴点は除去されます。

$$\{(XA - XB)^2 + (YA - YB)^2 \} \times del2\_bifurcation < del2\_distance$$



第二除去では、除去処理において分岐点の削除を抑制することが出来ます。



距離による除去を行わない場合は、del2_distance に 0 を指定します。
del2_distance に 0 を指定した場合は、第二除去の処理は行いません。

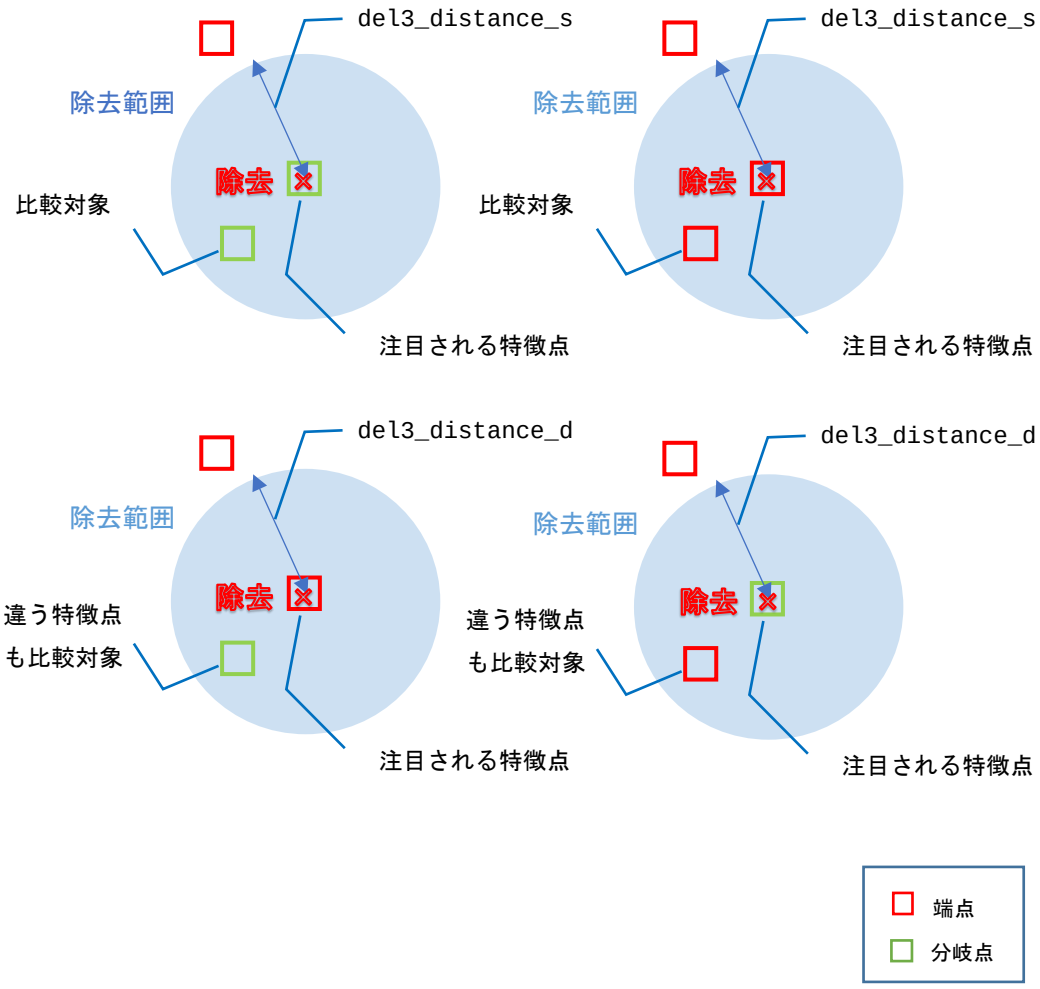
【第三除去】
第三除去は、第一除去、第二除去を行った後に除去処理を行います。
除去の条件は第一除去と同じですが、特徴点同士（違う種類も含めて）が近くにいる場合にも除去を行います。注目される特徴点（座標 XA、座標 YA）が分岐の場合は del3_bifurcation による除去抑制が働きます。比較対象の特徴点は（座標 XB、座標 YB）とします。

特徴点が同一の種別の場合は、del3_distance_s が以下の式を満たす場合は、除去判定されます。（注目ピクセルが端点の場合は、del3_bifurcation=1）

$$\{(XA - XB)^2 + (YA - YB)^2 \} \times del3\_bifurcation < del3\_distance\_s$$

特徴点が異なる種別の場合は、del3_distance_d が以下の式を満たす場合は、除去判定がされます。（注目ピクセルが端点の場合は、del3_bifurcation=1）

$$\{(XA - XB)^2 + (YA - YB)^2 \} \times del3\_bifurcation < del3\_distance\_d$$



第三除去は、第一除去と同様に信用度が低い場合の除去も行われます。これは del3_probability と del3_bifurcation が使われ、条件は第一除去と同様です。

距離による除去を行わない場合は、del3_distance_s と del3_distance_d に 0 を指定します。
信用度による除去を行わない場合は、 del3_probability に 0 を指定します。
del3_distance_s と del3_distance_d に 0 を指定し del3_probability に 0 を指定した場合は、第三除去の処理は行いません。

第一除去、第二除去、第三除去をすべて行わない設定は禁止とします。

注意	なし

4.11.6 CircleFitting

CircleFitting

円を検出します

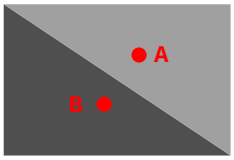
コンフィグレーションデータファイル		r_drp_circle_fitting.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		177312	
ヘッダファイル		r_drp_circle_fitting.h	
パラメータ	構造体名		
	r_drp_circle_fitting_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力データのアドレス
	src_width	uint16_t	入力画像の幅（ピクセル）
	src_height	uint16_t	入力画像の高さ（ピクセル）
	work	uint32_t	ワークエリアのアドレス
	c_area_startx	uint16_t	円の中心の探索領域の開始位置の X 座標
	c_area_starty	uint16_t	円の中心の探索領域の開始位置の Y 座標
	c_area_width	uint16_t	円の中心の探索領域の幅（ピクセル）
	c_area_height	uint16_t	円の中心の探索領域の高さ（ピクセル）
	min_radius	uint16_t	円の半径の最小値（2～478） （step 値よりも大きい値を設定してください）
	max_radius	uint16_t	円の半径の最大値（2～478） （min_radius 以上の値を設定してください）
	step	uint8_t	X 方向、Y 方向、半径方向の探索実行単位（ピクセル） （1～51）
入出力詳細	入力画像	アドレス ： src で指定 （dst、work と異なるアドレスにしてください） 幅（ピクセル） ： src_width で指定（16～1280） 高さ（ピクセル） ： src_height で指定（16～960） フォーマット ： 8bit グレイスケール（1 ピクセルあたり 1 バイト） データサイズ ： (src_width) × (src_height) × 1 バイト	
	探索領域	開始位置の X 座標 ：c_area_startx で指定 （min_radius + step ～ src_width - 1 - min_radius - step） 開始位置の Y 座標 ：c_area_starty で指定 （min_radius + step ～ src_height - 1 - min_radius - step） 幅（ピクセル） ：c_area_width で指定 （1～src_width - c_area_startx - min_radius - step） 高さ（ピクセル） ：c_area_height で指定 （1～src_height - c_area_starty - min_radius - step） ＜内容＞ 入力画像中の円の中心を探索する領域です。 c_area_startx + c_area_width の値が min_radius + step + 1 以上 src_width - min_radius - step 以下の値、 c_area_starty + c_area_height の値が min_radius + step + 1 以上 src_height - min_radius - step 以下の値となるように設定してください。詳細は解説を参照してください。	

	出力データ	アドレス	: dst で指定 （src、work と異なるアドレスにしてください）
		フォーマット	: アドレス先頭から順に、 発見した円の中心の X 座標値（2 バイト）、 発見した円の中心の Y 座標値（2 バイト）、 発見した円の半径（2 バイト）、 発見した円の score（2 バイト） （詳細は解説を参照してください）
		データサイズ	: 8 バイト
	ワークエリア	アドレス	: work で指定 （src、dst と異なるアドレスにしてください）
		データサイズ	: (c_area_width) × ((c_area_height ÷ step)【小数点以下切り上げ】) × 6 バイト
		＜内容＞ サークルフィッティング処理途中のデータを保存します。	
タイル数	2		
分割処理	不可 ただし CPU 処理と組み合わせる事で分割処理が可能です。 詳細は解説を参照してください。		

解説

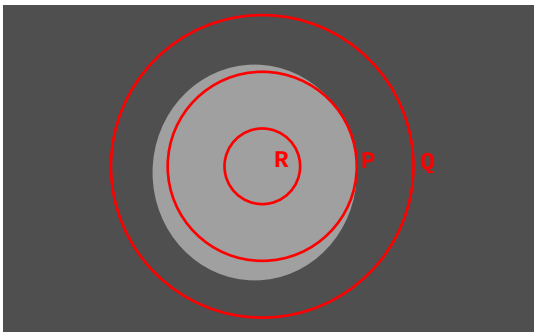
本機能は、src で指定したアドレスの画像に対して、サークルフィッティング処理を行い、dst で指定したアドレスに、発見した円の中心座標と半径と score を出力します。

輪郭が 1 つあると認識できるような画像においては、画像上のある点 A と、A と輪郭をはさんで反対側にある点 B には、輝度の差があります。



斜線の輪郭が1つある画像

サークルフィッティング処理では、上記の考えと、探す輪郭は円であることを利用し、ある円 P における、半径の少し大きい同心円 Q の輝度と、半径の少し小さい同心円 R の輝度の差の絶対値を計算します。



サークルフィッティング処理での計算対象

中心座標(x,y)、半径 r の円における、輝度を取得する箇所については、半径 r の円周上の 48 点(点(x+r,y)を起点として、7.5°ずつ取得箇所をずらします)となります。取得箇所の座標が整数でない場合には、小数点以下四捨五入により整数にしています。

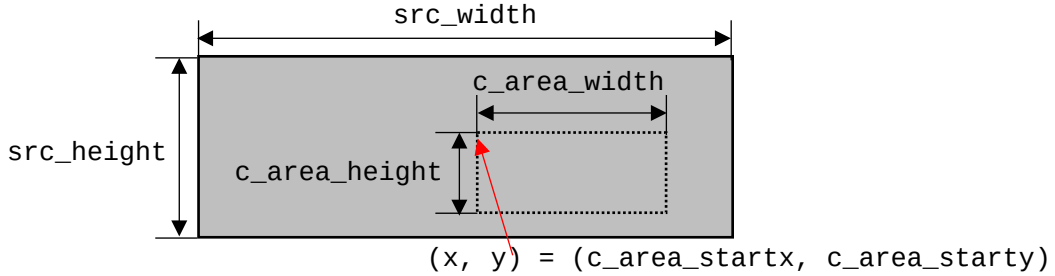
ある中心座標(x,y)、半径 r における、サークルフィッティング処理結果 score は、

$$\text{score} = \left|\left(\text{中心座標}(x,y)\text{で半径}(r+\text{step})\text{の円周上の 48 点の輝度値の合計}\right) - \left(\text{中心座標}(x,y)\text{で半径}(r-\text{step})\text{の円周上の 48 点の輝度値の合計}\right)\right|$$

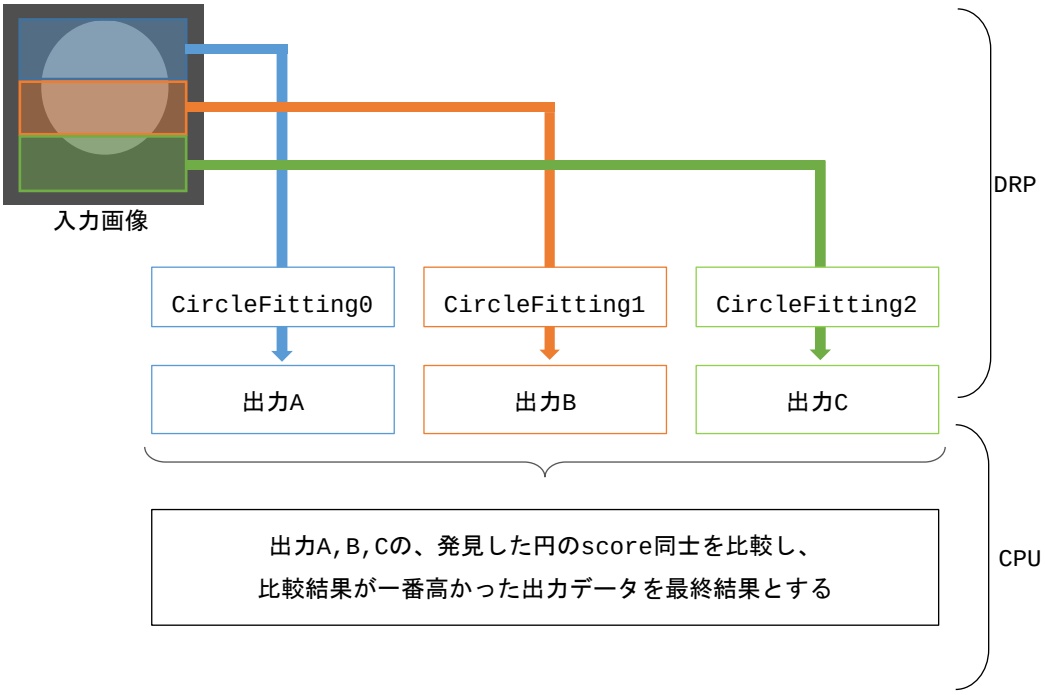
で計算されます。中心座標と半径を変えて計算し、この score が最も高い、円の中心の X 座標値、Y 座標値、半径 r、score を出力します。
なお、最も高い score が複数あった場合、優先度は下記の通りとなります。

1. 半径がより小さい
2. 中心の Y 座標値がより小さい
3. 中心の X 座標値がより小さい

円の中心の探索領域については、c_area_startx、c_area_starty、c_area_width、c_area_height により、下図のように決まります。円の中心の探索領域は入力画像をはみ出ないように設定してください。
下図の探索領域の内、中心座標(c_area_startx + step * n, c_area_starty + step * n)【n は 0 以上の整数】のサークルフィッティング処理を実施しますが、円が入力画像の外にはみ出る個所がある場合は、円として判定しません。



本機能は、CPU による分割処理を行う事が出来ます。
3 並列処理を行う例を以下に示します。
分割前と同じ中心座標のサークルフィッティング処理を実施するように、探索領域を 3 つの領域に分割し、CircleFitting0、CircleFitting1、CircleFitting2 に対して、それぞれ所定の dst、work、c_area_startx、c_area_starty、c_area_width、c_area_height を指定します。src、src_width、src_height、min_radius、max_radius、step は同じ設定としてください。
DRP によるサークルフィッティング処理が完了した後、CircleFitting0、CircleFitting1、CircleFitting2 の dst 領域に出力された、発見した円の score 同士を比較し、比較結果が一番高かった出力データを最終結果とする事で、分割処理を実現する事が可能です。



注意	探索領域のどの点を中心としても、円の一部または全部が入力画像外となるパラメータ設定を行ったときは、出力データは全て 0 となります。

4.11.7 FindContours

FindContours

輪郭を検出し、その外接矩形を算出します

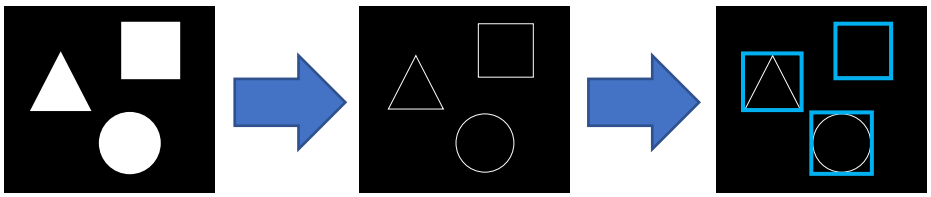
コンフィグレーションデータファイル			r_drp_find_contours.dat
対応バージョン			1.01
コンフィグレーションデータサイズ（バイト）			204640
ヘッダファイル			r_drp_find_contours.h
パラメータ	構造体名		
	r_drp_find_contours_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst_rect	uint32_t	出力データ（矩形情報）のアドレス
	dst_region	uint32_t	出力データ（領域情報）のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	work	uint32_t	ワークエリアのアドレス
	dst_rect_size	uint32_t	矩形情報の最大出力個数（1～20,000）
	dst_region_size	uint32_t	領域情報の最大出力個数（0～500,000） 輪郭毎の個数ではなく、出力される領域情報の総数の上限となる値を指定してください。 （0 を指定した場合は、領域情報は出力されません）
	threshold_width	uint16_t	検出対象とする矩形の幅の閾値（1～width）
	threshold_height	uint16_t	検出対象とする矩形の高さの閾値（1～height）
入出力詳細	入力画像	アドレス	： src で指定
		幅（ピクセル）	： width で指定（64～1280、8 の整数倍）
		高さ（ピクセル）	： height で指定（32～960）
		フォーマット	： 二値画像（1 ピクセルあたり 1 バイト）または 8bit グレイスケール（1 ピクセルあたり 1 バイト） （詳細は解説を参照してください）
		データサイズ	： (width) × (height) × 1 バイト
	出力データ（矩形情報）	アドレス	： dst_rect で指定 （src と異なるアドレスにしてください）
		フォーマット	： アドレス先頭から順に、 輪郭の外接矩形の左上の X 座標値（2 バイト）、 輪郭の外接矩形の左上の Y 座標値（2 バイト）、 輪郭の外接矩形の幅（2 バイト）、 輪郭の外接矩形の高さ（2 バイト）、 領域情報の個数（4 バイト）、 領域情報の先頭アドレス（4 バイト） （詳細は解説を参照してください）
		終端データ	：矩形情報の終端、全フィールドが 0（16 バイト） （詳細は解説を参照してください）
		データサイズ	：（検出された矩形の個数 + 1）× 16 バイト 「+ 1」は終端データ分のサイズです。 最大で（dst_rect_size）× 16 バイトとなります。
	出力データ（領域情報）	アドレス	： dst_region で指定 （src と異なるアドレスにしてください）
		フォーマット	： アドレス先頭から順に、 輪郭を構成するピクセルの X 座標値（2 バイト）、 輪郭を構成するピクセルの Y 座標値（2 バイト） （詳細は解説を参照してください）
		データサイズ	：（全ての輪郭を構成するピクセルの総数） × 4 バイト 最大で（dst_region_size）× 4 バイトとなります。

	ワークエリア	アドレス : work で指定 データサイズ : (width) × (height) × 1 バイト ＜内容＞ 輪郭検出処理途中のデータを保存します。
タイル数	2	
分割処理	不可	

解説

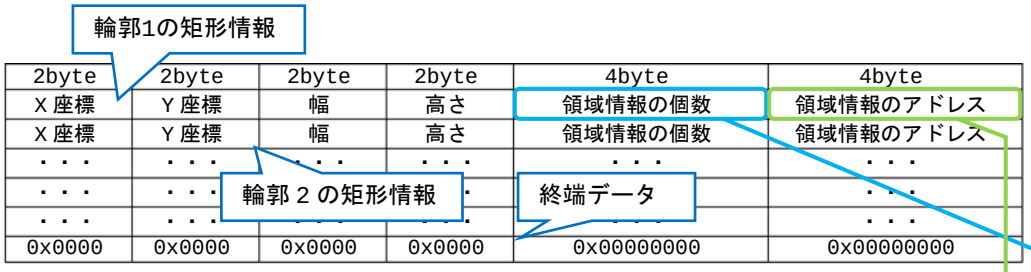
本機能は、src で指定されたアドレスの画像に対して、輪郭検出処理を行い、dst_rect に発見した輪郭の外接矩形に関する情報を、dst_region で指定したアドレスに発見した輪郭を構成するピクセルの座標を出力します。

左下のような画像を入力とした場合には、真ん中下の画像のように 3 つ輪郭を検出します。また、検出したそれぞれの輪郭に対して、右下の画像のような外接矩形を算出し、その情報を「矩形情報」として出力し、輪郭を構成するピクセルの座標を「領域情報」として出力します。

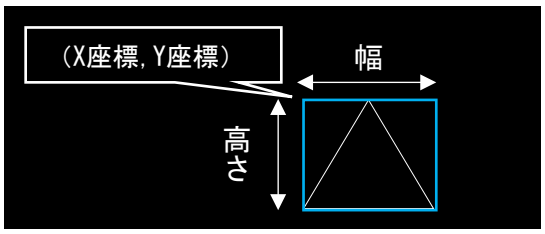


入力画像から矩形を検出した模式図

「矩形情報」
入力画像上で発見された輪郭の外接矩形の情報及び、該当する領域情報へのポインタとその個数が下図のように出力されます。このデータセットが、検出された輪郭の個数分出力された後に、データの終端を表す「終端データ」が出力されます。この終端データが読み出せるまで、矩形情報を読み進めることで、全ての矩形情報を取得できます。また、領域情報の個数とアドレスを使用することで、対応する領域情報を取得できます。

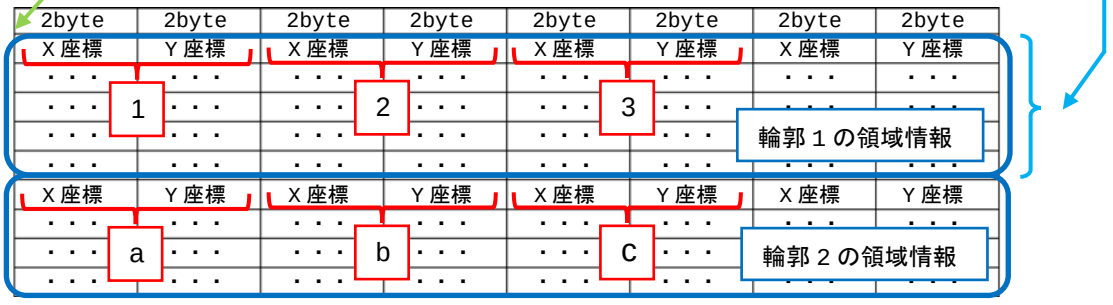


矩形情報

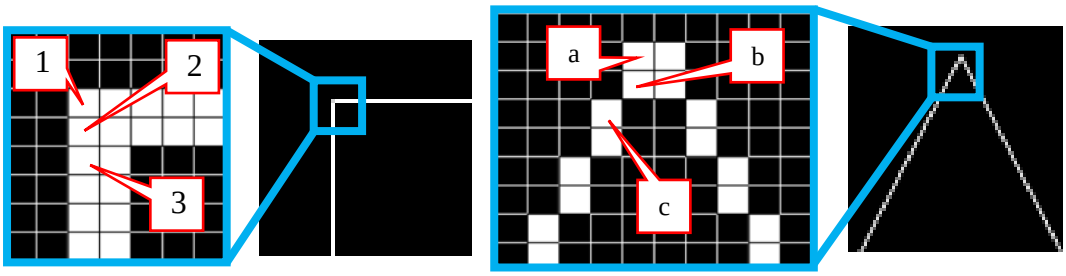


矩形情報の説明

「領域情報」
輪郭を構成する全てのピクセルの（X,Y）座標が、下図のように輪郭ごとに出力されます。この座標は、入力画像の一番左上のピクセルを(0,0)とする座標系で表されています。



領域情報



輪郭1 輪郭2

データの出力個数が dst_rect_size または dst_region_size に設定した値に達した場合、上限に達した方のデータ出力は停止しますが、もう一方のデータは引き続き出力されます。両方のデータ個数が上限に達した場合は、データ出力を完全に停止して DRP は処理を終了します。また、終端データの出力前に、矩形情報の出力個数が上限に達した場合は、終端データは出力されません。

本機能は、入力画像のフォーマットとして二値画像を想定しています。8bit グレイスケールが入力された場合は、画素値が 1 以上のピクセルを画素値が 1 のピクセル(輪郭)として扱い、輪郭検出を実行します。

本機能は、パラメータの threshold_width,threshold_height に値を設定することで、検出した矩形の幅か高さが設定値より小さい場合は、その輪郭の「領域情報」と「矩形情報」を出力から除外します。

本機能は OpenCV の cv::findContours 関数の引数 mode に CV_RETR_LIST、method に CV_CHAIN_APPROX_NONE を指定した場合と同等の処理を実行します。ただし、出力フォーマットは独自のものとなっています。

参考 URL： https://opencv.org/

本機能は、src と work に同一アドレスを指定することが可能です。しかし、本機能は work で指定した領域に処理途中のデータを書き出しながら処理を進めるため、同一のアドレスを指定した場合は入力画像が破壊されます。

注意	本機能の出力サイズは入力画像によって異なります。メモリ破壊を避けるために、入出力詳細の矩形情報と領域情報を参照して、dst_rect_size と dst_region_size に適切な値を設定し、dst_rect と dst_region にはデータサイズ以上のメモリ領域を割り当ててください。

4.12 Histograms

4.12.1 Histogram

Histogram

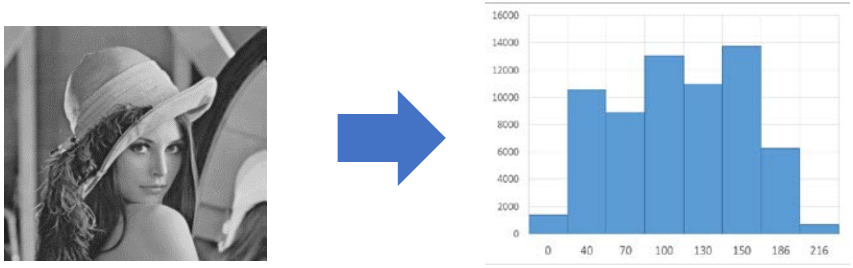
入力画像のヒストグラムを生成します

コンフィグレーションデータファイル		r_drp_histogram.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		83424	
ヘッダファイル		r_drp_histogram.h	
パラメータ	構造体名		
	r_drp_histogram_t		
	メンバ名	型	説明
	src	uint32_t	入力データのアドレス
	dst	uint32_t	出力データのアドレス
	data_size	uint32_t	入力データ数（バイト）
	mask	uint32_t	マスクデータのアドレス
	ranges	uint32_t	ヒストグラムのビンの幅指定領域のアドレス
	hist_size	uint16_t	ヒストグラムのビンの個数
	accumulate	uint8_t	累積フラグ（0:初期化、1:累積）
入出力詳細	入力データ	アドレス	: src で指定 （dst、mask、ranges と異なるアドレスにしてください）
		データ数	: data_size で指定（256～1,228,800）
		フォーマット	: 8bit（1 データ辺り 1 バイト）
		データサイズ	: data_size × 1 バイト
	出力データ	アドレス	: dst で指定 （src、mask、ranges と異なるアドレスにしてください）
		ビンの個数	: hist_size で指定（1～256）
		フォーマット	: 度数（1 ビン辺り 4 バイト） accumulate の設定が累積の時は、dst で指定された領域の度数が読み出されて各ビンの初期値となります。 uint32_t の最大値を超えた場合は、最大値のままになります。 詳細は解説を参照してください。
		データサイズ	: hist_size × 4 バイト
	ビン指定	アドレス	: ranges で指定 （src、dst、mask と異なるアドレスにしてください）
		ビン範囲の個数	: hist_size + 1
		フォーマット	: 16bit（0～256） 0 番目のビンの下限を ranges で指定したアドレス+0（バイト）に設定します。0 番目のビンの上限を ranges で指定したアドレス+2（バイト）に設定します。 1 番目のビンの下限は ranges で指定したアドレス+2（バイト）に設定した値になります。
		データサイズ	: （ hist_size + 1 ） × 2 バイト
			＜内容＞ 全てのビンの下限・上限を設定します。i 番目のビンは、ranges で指定したアドレス+i×2（バイト）に設定された値以上で、ranges で指定したアドレス+i×2+2（バイト）に設定された値未満になります。 ranges で設定する値の個数は、hist_size+1 個の値を設定して下さい。 詳細は解説を参照してください。

	マスクデータ	アドレス	: mask で指定 （src、dst、ranges と異なるアドレスにしてください） mask に 0 を指定するとマスク機能は無効になります。
		データ数	: 入力データと同じ
		フォーマット	: 8bit（1 データ辺り 1 バイト） 0 以外の値を指定された場合のみヒストグラムをカウントします
		データサイズ	: 入力データと同じ
		＜内容＞ マスクデータが 0 以外の値を指定されている入力データのヒストグラムをカウントします。 詳細は解説を参照してください。	
タイル数	2		
分割処理	不可 ただし CPU 処理と組み合わせる事で分割処理が可能です。 詳細は解説を参照してください。		

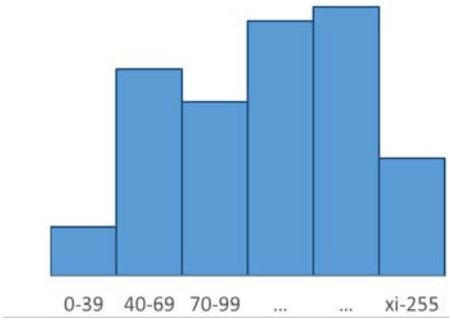
解説

本機能は、srcで指定したアドレスのデータのヒストグラムの算出を行い、dstで指定したアドレスに出力します。data_sizeに画像のデータサイズ（＝幅×高さ）を指定する事で、以下のように画像の入力が可能になります。



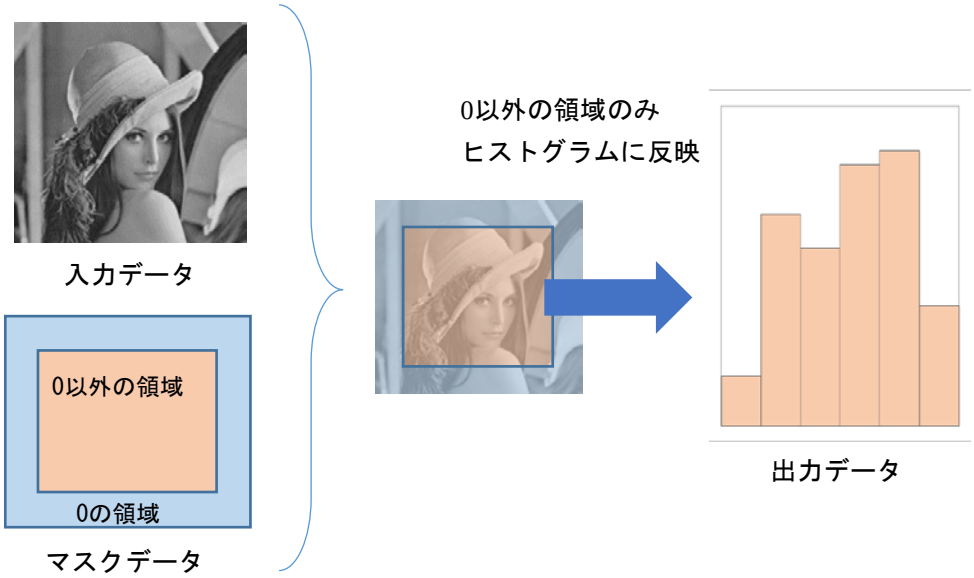
入力データ　出力データ

本機能は、hist_size、rangesを使用してビンの範囲を設定できます。
hist_size個のビンの上限と下限を設定するために、hist_size+1個のビンの範囲を指定します。
i番目のビンの下限はアドレスrange+i×2に設定します。
i番目のビンの上限はアドレスrange+(i+1)×2に設定します。
以下にi+1個のビン指定を行う場合の例を示します。例ではi番目のビンはx_iを下限、255を上限として設定しています。

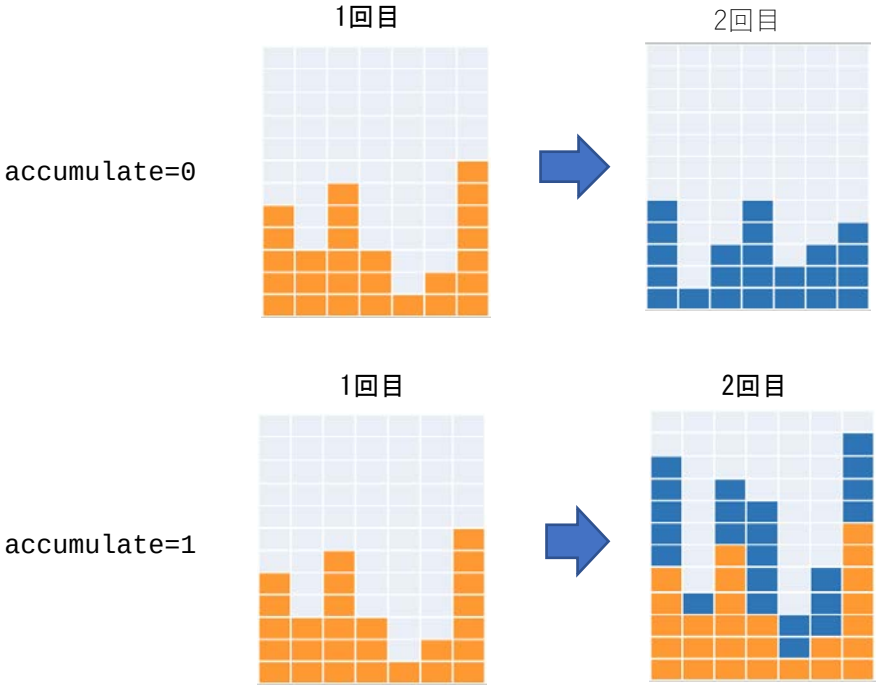


アドレス	ビン指定	
ranges+0	0	0番目のビン
ranges+2	40	1番目のビン
ranges+4	70	2番目のビン
ranges+6	100	
:	:	
ranges+i×2	x_i	i番目のビン
ranges+(i+1)×2	256	

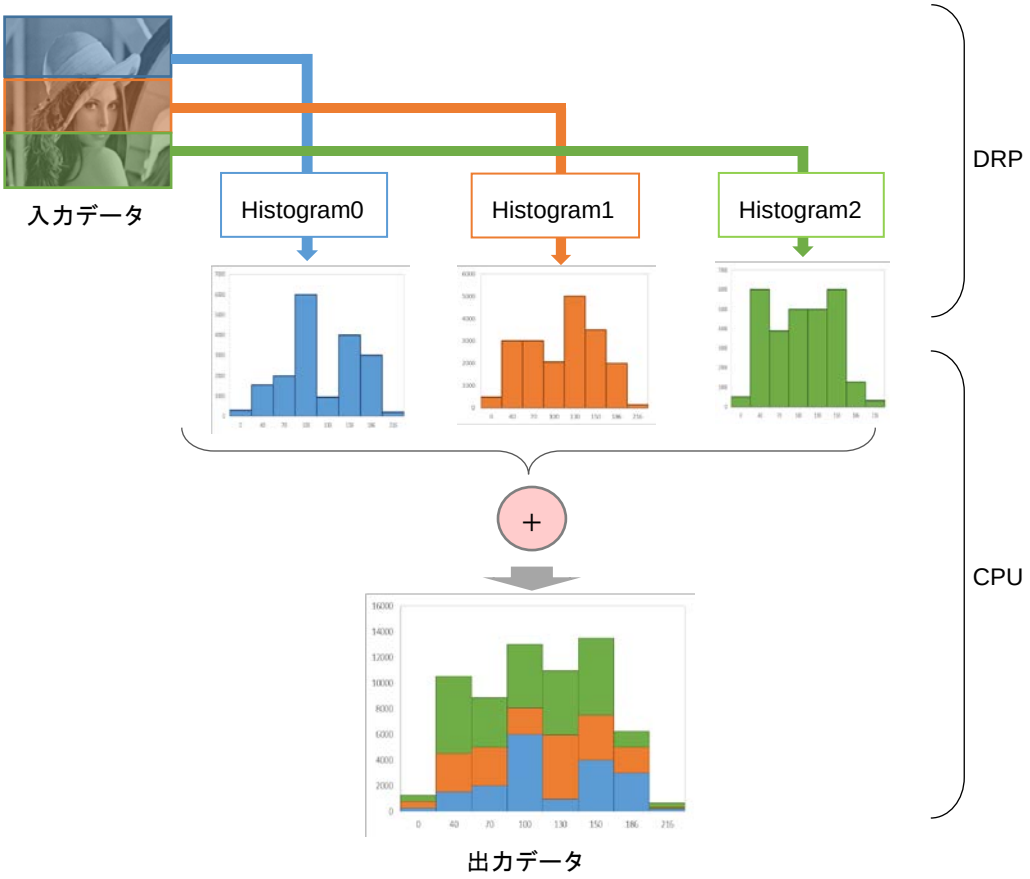
本機能は、maskを使用してヒストグラムをカウントするデータをマスクする事が可能です。
マスクデータに0の値が入っている領域のデータはヒストグラムとしてカウントされず、0以外の値が入っている領域のみヒストグラムとしてカウントされます。



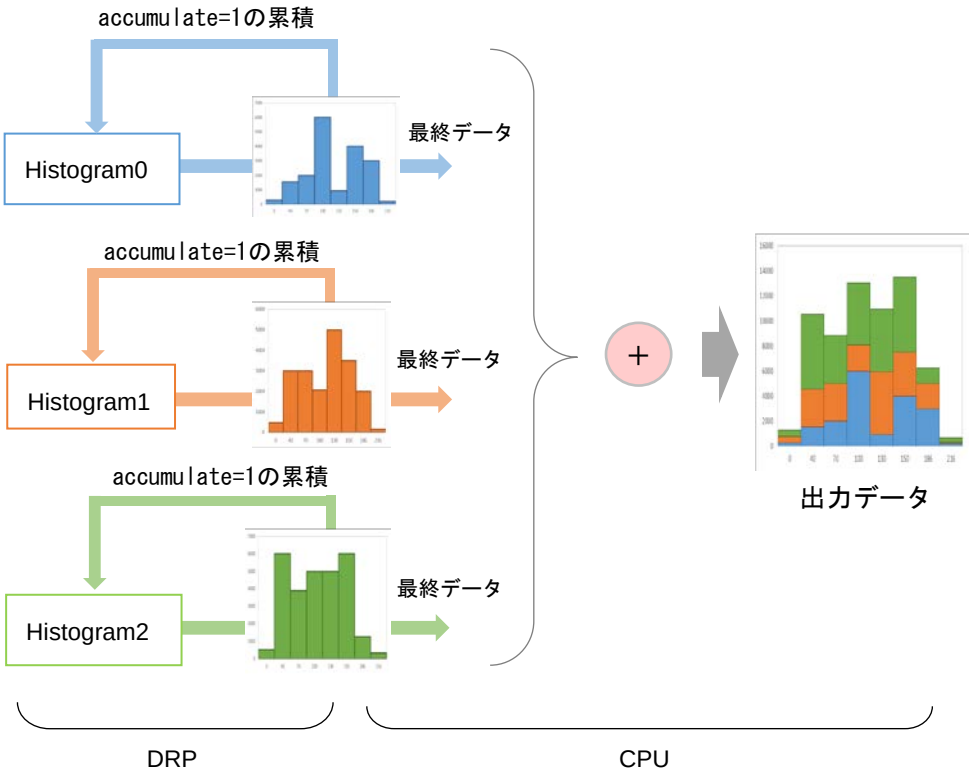
本機能は、accumulateを使用して、ヒストグラムの値の初期値と累積を選択する事が可能です。accumulateに1を指定すると、dstで指定したアドレスのヒストグラムの結果を読み込んで初期値とします。accumulateに0を指定すると、ヒストグラムの初期値は全て0になります。
従って累積を行う場合は、ビン指定（hist_size、ranges）を変更できません。また累積によって度数が4,294,967,295（=2^{32}-1）を超える場合は、4,294,967,295に留まります。



本機能は、CPUによる分割処理を行う事が出来ます。
accumulate=0の設定において、3並列処理を行う例を以下に示します。
入力データを3つの領域に分割し、Histogram0、Histogram1、Histogram2に対して、それぞれ所定のsrc、dst、mask（必要に応じて）、data_sizeを指定します。rangesやhist_sizeは同じ設定として下さい。
DRPによるヒストグラムの算出が完了した後、Histogram0、Histogram1、Histogram2のdst領域の各ビンの度数をCPUにより加算する事で分割処理を実現する事が可能です。



accumulate=1の設定において、3並列処理を行う例を以下に示します。
accumulate=1の設定時は、accumulateによる累積がすべて完了した後にCPUによってdst領域の各ビンの度数を加算する事で分割処理を実現する事が可能です。



本機能は、OpenCV の cv::calcHist 関数の引数 nimages に 1、channels[]に{0}、dims に 1、uniform に false を指定した場合と同等の結果が得られます。

参考URL： https://opencv.org/

注意	なし

4.12.2 HistogramNormalization

HistogramNormalization

画像をヒストグラム正規化します

コンフィグレーションデータファイル			r_drp_histogram_normalization.dat
対応バージョン			1.01
コンフィグレーションデータサイズ（バイト）			45376
ヘッダファイル			r_drp_histogram_normalization.h
パラメータ	構造体名		
	r_drp_histogram_normalization_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力データ/出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	mode	uint8_t	1:MODE1 に設定（画像全体の明るさを調査する） 2:MODE2 に設定（画像を正規化する） （詳細は解説を参照してください）
	以下は MODE2 用のパラメータです。（MODE1 の場合は、0 を指定してください）		
	src_pixel_mean	uint32_t	入力画像の画素値の平均 上位 20bit が整数部、下位 12bit が小数部 （詳細は解説を参照してください）
	src_pixel_rstd	uint32_t	入力画像の画素値の標準偏差の逆数 上位 20bit が整数部、下位 12bit が小数部 （詳細は解説を参照してください）
	dst_pixel_mean	uint8_t	出力画像の画素値の平均
	dst_pixel_std	uint8_t	出力画像の画素値の標準偏差
入出力詳細	入力画像	アドレス	： src で指定
		幅（ピクセル）	： width で指定（8～1280、8 の整数倍）
		高さ（ピクセル）	： height で指定（8～960）
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (width) × (height) × 1 バイト
	出力データ (MODE1)	アドレス	： dst で指定
		フォーマット	： アドレス先頭から順に、 画素値の和（8 バイト） 画素値の二乗和（8 バイト）
		データサイズ	： 16 バイト
	出力画像 (MODE2)	アドレス	： dst で指定
		幅（ピクセル）	： 入力画像と同じ
		高さ（ピクセル）	： 入力画像と同じ
		フォーマット	： 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	： (width) × (height) × 1 バイト
タイル数	1		
分割処理	可 本機能は CPU 処理と組み合わせる事で分割処理が可能です。 詳細は解説を参照してください。		

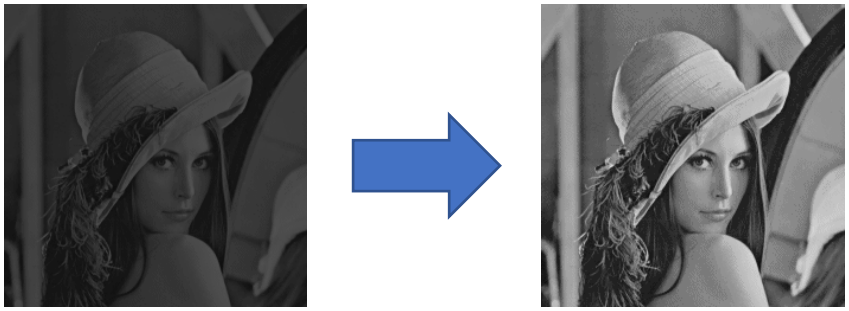
解説

本機能には、以下のような 2 つの動作モードがあり、組み合わせて使用することで、入力画像をヒストグラム正規化した出力画像を取得できます。

MODE1：src で指定された入力画像の画素値の和と二乗和を計算し、dst で指定したアドレスに出力します。
MODE2：src で指定された入力画像を、以下の計算によってヒストグラム正規化し、dst で指定したアドレスに出力します。

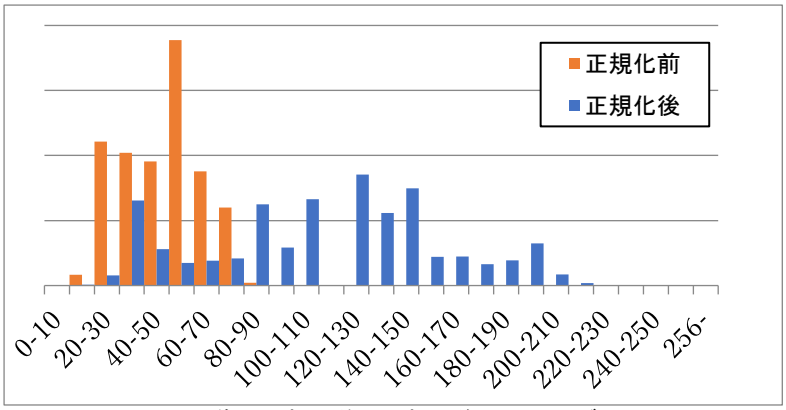
$$出力画素値 = \{(入力画素値 - \mathrm{src\_pixel\_mean}) \times \mathrm{src\_pixel\_rstd}\} \times \mathrm{dst\_pixel\_std} + \mathrm{dst\_pixel\_mean}$$

dst_pixel_mean と dst_pixel_std には、正規化後の画像の平均と標準偏差にしたい値を設定してください。左下の入力画像を dst_pixel_mean=112, dst_pixel_std=48 でヒストグラム正規化すると右下のような出力画像が得られます。



入力画像　出力画像

また、下図は上図の入力画像（正規化前）と出力画像（正規化後）のヒストグラムです。



入力画像の正規化前と正規化後のヒストグラム

下記の手順で、入力画像のヒストグラムを正規化した出力画像を取得できます。
1. MODE1 を実行し、画素値の和と二乗和を算出します。
2. 1.の出力結果と下記の式から、src_pixel_mean と src_pixel_rstd を CPU で計算します。
3. 2.の計算結果をパラメータとして MODE2 を実行することで、正規化された画像が出力されます。

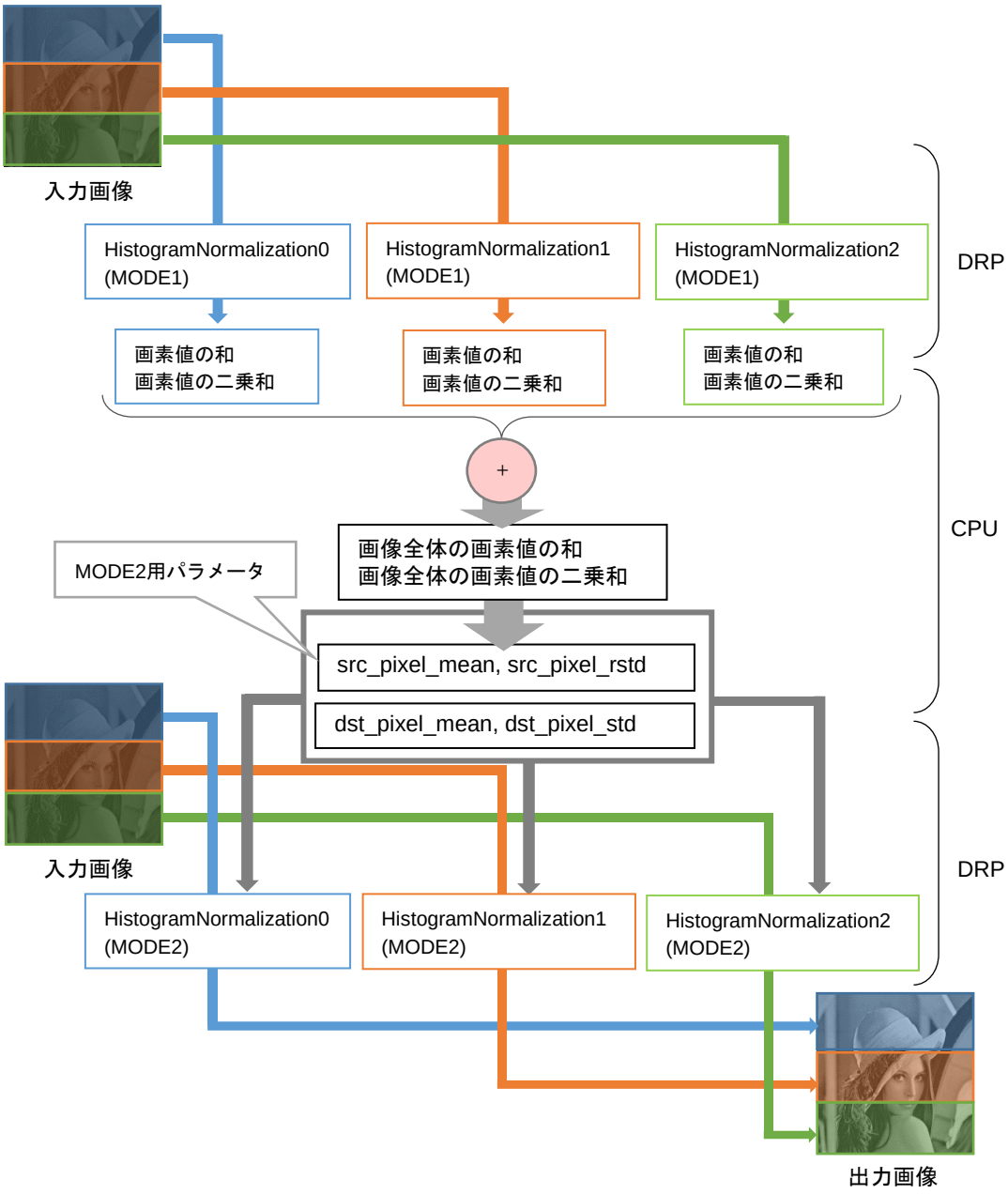
$$\mathrm{src\_pixel\_mean} = 画素値の和 \div (\mathrm{width} \times \mathrm{height}) \times 4096$$

$$\mathrm{src\_pixel\_rstd} = 1 \div \left(\sqrt{画素値の二乗和 \div (\mathrm{width} \times \mathrm{height}) - \left(画素値の和 \div (\mathrm{width} \times \mathrm{height})\right)^2}\right) \times 4096$$

src_pixel_mean と pixel_rstd は、固定小数点（上位 20bit が整数部、下位 12bit が小数部）であるため、上の式のように必ず×4096 してください。

本機能は、CPU 処理と組み合わせる事で分割処理が可能です。3 並列処理の例を以下に示します。

1. 入力画像を 3 つの領域に分割し、各タイルの HistogramNormalization に対して、それぞれ所定の src、dst、width、height を、mode には 1 を指定してください。
2. DRP による処理が完了した後、各 HistogramNormalization の dst 領域の画素値の和と二乗和を CPU で足し合わせた値を使って、src_pixel_mean と src_pixel_rstd を計算してください。
3. 入力画像を 3 つの領域に分割し、各タイルの HistogramNormalization に対して、それぞれ所定の src、dst、width、height を指定してください。また、共通のパラメータとして、src_pixel_mean と src_pixel_rstd には 2.の計算結果を、dst_pixel_mean, dst_pixel_std には任意の値を、mode には 2 を指定してください。
4. DRP による処理が完了すると、ヒストグラム正規化された画像が出力されます。



本機能は、動作モードが MODE2 の場合、src と dst に同一アドレスを指定することが可能です。

注意　誤動作の原因となるため、src_pixel_mean と src_pixel_rstd には、解説に記載している式で算出した値以外は設定しないでください。

4.12.3 HistogramNormalizationRgb

HistogramNormalizationRgb

画像（RGB）をヒストグラム正規化します

コンフィグレーションデータファイル		r_drp_histogram_normalization_rgb.dat	
対応バージョン		1.01	
コンフィグレーションデータサイズ（バイト）		60960	
ヘッダファイル		r_drp_histogram_normalization_rgb.h	
パラメータ	構造体名		
	r_drp_histogram_normalization_rgb_t		
	メンバ名	型	説明
	src	uint32_t	入力画像のアドレス
	dst	uint32_t	出力データ/出力画像のアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	mode	uint8_t	1:MODE1 に設定（画像全体の明るさを調査する） 2:MODE2 に設定（画像を正規化する） （MODE の詳細は解説を参照してください）
	以下は MODE2 用のパラメータです。（MODE1 の場合は、0 を指定してください）		
	src_pixel_red_mean	uint32_t	入力画像の R 成分の画素値の平均 上位 20bit が整数部、下位 12bit が小数部 （詳細は解説を参照してください）
	src_pixel_red_rstd	uint32_t	入力画像の R 成分の画素値の標準偏差の逆数 上位 20bit が整数部、下位 12bit が小数部 （詳細は解説を参照してください）
	src_pixel_green_mean	uint32_t	入力画像の G 成分の画素値の平均 上位 20bit が整数部、下位 12bit が小数部 （詳細は解説を参照してください）
	src_pixel_green rstd	uint32_t	入力画像の G 成分の画素値の標準偏差の逆数 上位 20bit が整数部、下位 12bit が小数部 （詳細は解説を参照してください）
	src_pixel_blue_mean	uint32_t	入力画像の B 成分の画素値の平均 上位 20bit が整数部、下位 12bit が小数部 （詳細は解説を参照してください）
	src_pixel_blue_ rstd	uint32_t	入力画像の B 成分の画素値の標準偏差の逆数 上位 20bit が整数部、下位 12bit が小数部 （詳細は解説を参照してください）
	dst_pixel_mean	uint8_t	出力画像の画素値の平均
	dst_pixel_std	uint8_t	出力画像の画素値の標準偏差
入出力詳細	入力画像	アドレス : src で指定	
		幅（ピクセル） : width で指定（8～1280、8 の整数倍）	
		高さ（ピクセル） : height で指定（8～960）	
		フォーマット : RGB（1 ピクセルあたり 3 バイト）	
		データサイズ : (width) × (height) × 3 バイト	
	出力データ (MODE1)	アドレス : dst で指定	
		フォーマット : アドレス先頭から順に、 R 成分の画素値の和（8 バイト） R 成分の画素値の二乗和（8 バイト） G 成分の画素値の和（8 バイト） G 成分の画素値の二乗和（8 バイト） B 成分の画素値の和（8 バイト） B 成分の画素値の二乗和（8 バイト）	
		データサイズ : 48 バイト	

	出力画像 (MODE2)	アドレス 幅（ピクセル） 高さ（ピクセル） フォーマット データサイズ	: dst で指定 : 入力画像と同じ : 入力画像と同じ : RGB（1 ピクセルあたり 3 バイト） : (width) × (height) × 3 バイト
タイル数	1		
分割処理	可 本機能は CPU 処理と組み合わせる事で分割処理が可能です。 詳細は解説を参照してください。		

解説

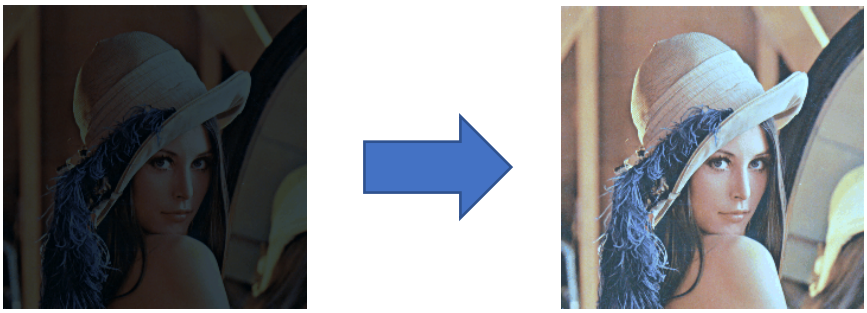
本機能には、以下のような 2 つの動作モードがあり、組み合わせて使用することで、入力画像をヒストグラム正規化した出力画像を取得できます。

MODE1：src で指定された入力画像の画素値の和と二乗和を計算し、dst で指定したアドレスに出力します。
MODE2：src で指定された入力画像を、以下の計算によってヒストグラム正規化し、dst で指定したアドレスに出力します。

$$出力画素値 = \{(入力画素値 - src\_pixel\_*\_mean) \times src\_pixel\_*\_rstd\} \times dst\_pixel\_std + dst\_pixel\_mean$$

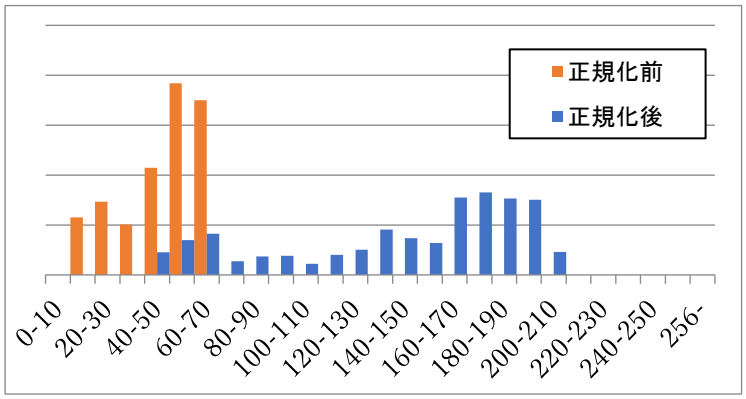
*には、入力画素値の成分によって、対応する red、green、blue のいずれかが入ります。

dst_pixel_mean と dst_pixel_std には、正規化後の画像の平均と標準偏差にしたい値を設定してください。左下の入力画像を dst_pixel_mean=144,dst_pixel_std=48 でヒストグラム正規化すると右下のような出力画像が得られます。



入力画像　　出力画像

また、下図は上図の入力画像（正規化前）と出力画像（正規化後）の R 成分のヒストグラムです。



入力画像の正規化前と正規化後の R 成分のヒストグラム

下記の手順で、入力画像のヒストグラを正規化した出力画像を取得できます。
1. MODE1 を実行し、画素値の和と二乗和を算出します。
2. 1.の出力結果と下記の式から、src_pixel_*_mean と src_pixel_*_rstd を CPU で計算します。
3. 2.の計算結果をパラメータとして MODE2 を実行することで、正規化された画像が出力されます。

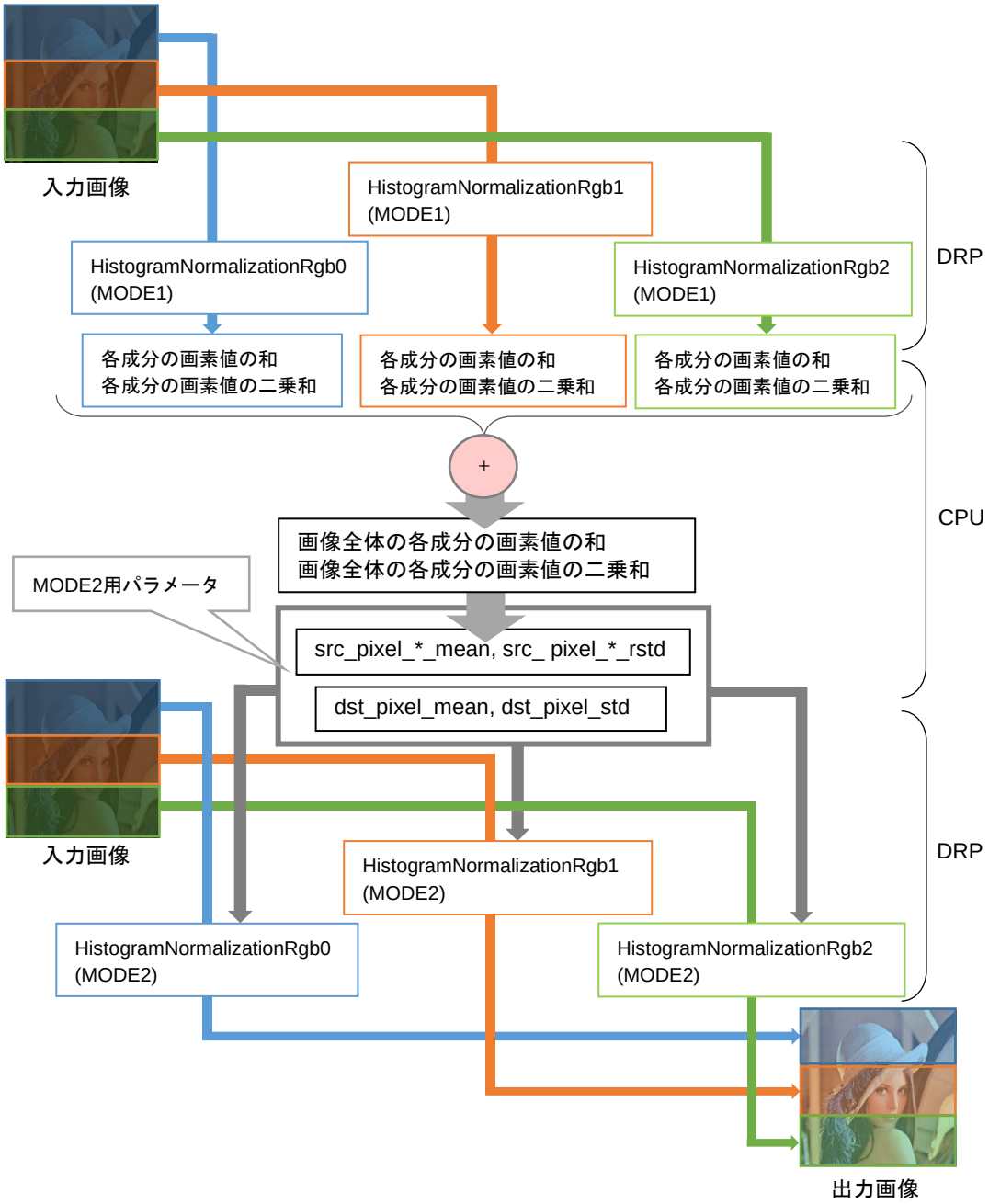
$$src\_pixel\_*\_mean = 画素値の和 \div (width \times height) \times 4096$$

$$src\_pixel\_*\_rstd = 1 \div \left(\sqrt{画素値の二乗和 \div (width \times height) - \left(画素値の和 \div (width \times height)\right)^2} \right) \times 4096$$

src_pixel_*_mean と src_pixel_*_rstd は、固定小数点（上位 20bit が整数部、下位 12bit が小数部）であるため、上の式のように必ず×4096 してください。

本機能は、CPU 処理と組み合わせる事で分割処理が可能です。3 並列処理の例を以下に示します。

1. 入力画像を 3 つの領域に分割し、各タイルの HistogramNormalizationRgb に対して、それぞれ所定の src、dst、width、height を、mode には 1 を指定してください。
2. DRP による処理が完了した後、各 HistogramNormalizationRgb の dst 領域の各成分の画素値の和と二乗和を CPU で足し合わせた値を使って、src_pixel_*_mean と src_pixel_*_rstd を計算してください。
3. 入力画像を 3 つの領域に分割し、各タイルの HistogramNormalizationRgb に対して、それぞれ所定の src、dst、width、height を指定してください。また、共通のパラメータとして、src_pixel_*_mean と src_pixel_*_rstd には 2.の計算結果を、dst_pixel_mean, dst_pixel_std には任意の値を、mode には 2 を指定してください。
4. DRP による処理が完了すると、ヒストグラム正規化された画像が出力されます。



本機能は、動作モードが MODE2 の場合、src と dst に同一アドレスを指定することが可能です。

注意　誤動作の原因となるため、src_pixel_*mean と src_pixel_*_rstd には、解説に記載している式で算出した値以外は設定しないでください。

4.13 Other

4.13.1 ReedSolomon

ReedSolomon

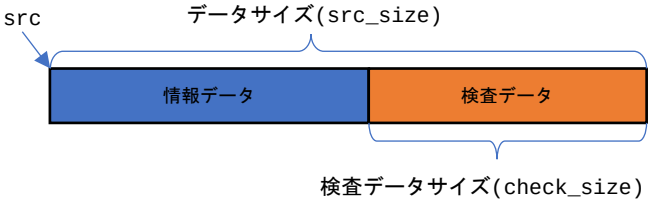
Reed-Solomon 符号を用いた誤り訂正をします（原始多項式固定）

コンフィグレーションデータファイル		r_drp_reed_solomon.dat	
対応バージョン		1.00	
コンフィグレーションデータサイズ（バイト）		118912	
ヘッダファイル		r_drp_reed_solomon.h	
パラメータ	構造体名		
	r_drp_reed_solomon_t		
	メンバ名	型	説明
	src	uint32_t	入力データのアドレス
	dst	uint32_t	出力データのアドレス
	src_size	uint16_t	入力データサイズ（バイト）
	check_size	uint16_t	検査データサイズ（バイト）

入出力詳細

入力データ

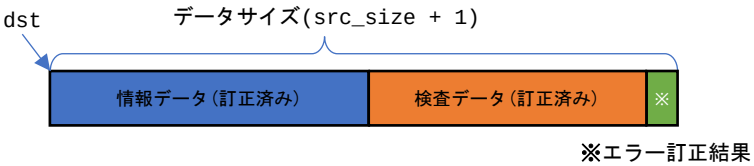
アドレス　：src で指定
データサイズ　：src_size で指定（2～254 バイト）
情報データ　：エラー訂正を行うデータ（1～253 バイト）
検査データ　：エラー訂正を行うためエンコードで付加された検査データ
検査データサイズ　：check_size で指定（1～127 バイト）



情報データ及び検査データは、リードソロモン符号エンコードした結果を入力します。
エンコード結果は、原始多項式の0次の係数がLSBであることを想定しています。

出力データ

アドレス　：dst で指定
データサイズ　：src_size + 1 バイト（エラー訂正結果）
情報データ（訂正済み）　：エラー訂正済みの情報データ（サイズは入力データの情報データと同一）
検査データ（訂正済み）　：エラー訂正済みの検査データ（サイズは入力データの検査データと同一）
エラー訂正結果　：エラー訂正結果を示すデータ（1 バイト）



タイル数	1
分割処理	不可

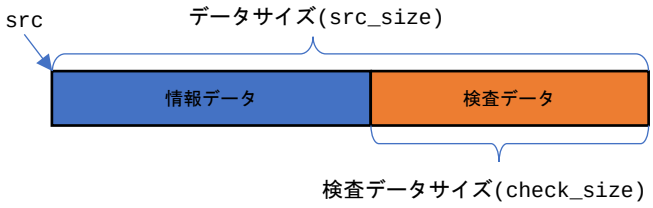
解説	本機能は、src で指定された入力データに対して、下記の仕様に対応したリードソロモン符号デコードを行い、dst で指定されたアドレスに訂正済みのデータとエラー訂正結果を出力します。任意の原始多項式を設定したい場合は、ReedSolomonGf8 機能を使用してください。 リードソロモン符号デコードの仕様： ・ガロアフィールド　　：GF(2^8) ・ガロア体の原始多項式　　：$\chi^8+\chi^4+\chi^3+\chi^2+1$ ・シンボルあたりのビット数：8 エラー訂正の結果は出力符号データの最後に付加される「エラー訂正結果」に格納されます。 エラー訂正に成功した場合は「0」、失敗した場合は「1」が格納されます。 エラー訂正可能なシンボル数は、floor(check_size ÷ 2)となります。従って、check_size に 1 を設定した場合、訂正は行われず出力データは変化しません。エラー訂正結果には 0 が出力されます。 本機能は、シンドローム算出、ユークリッド互除法、チェン探索、エラー値算出の順番でデコード処理を行います。エラーがない場合は、シンドローム算出で処理を停止します。エラーがある場合は、エラー値算出まで行いますが、エラーの数が多くなるにつれて、ユークリッド互除法とエラー値算出の処理時間が大きくなります。 従って、入力データのエラー数により処理時間が変動します。
注意	入力データ内のエラー数が訂正可能なシンボル数より多い場合は、誤訂正が発生することがあります。誤訂正とは不正なデコード処理を行う事で、エラー訂正に失敗しているのに「エラー訂正結果」が失敗（1）とならず、エラーではないシンボルを変更することもあります。

4.13.2 ReedSolomonGf8

ReedSolomonGf8

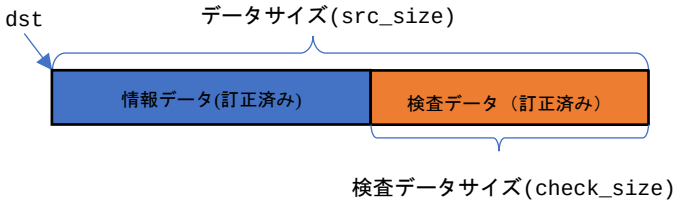
GF(2^8)の Reed-Solomon 符号を用いた誤り訂正をします

コンフィグレーションデータファイル			r_drp_reed_solomon_gf8.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			119872
ヘッダファイル			r_drp_reed_solomon_gf8.h
パラメータ	構造体名		
	r_drp_reed_solomon_gf8_t		
	メンバ名	型	説明
	src	uint32_t	入力データのアドレス
	dst	uint32_t	出力データのアドレス
	correct_addr	uint32_t	エラー訂正数を出力するアドレス
	src_size	uint16_t	入力データサイズ（バイト）
	check_size	uint16_t	検査データサイズ（バイト）
	primitive	uint16_t	ガロア体の原始多項式（0 次の係数を LSB とする） $\chi^8+\chi^4+\chi^3+\chi^2+1$ を設定する場合は 0x11D
入出力詳細	入力データ	アドレス	：src で指定
		データサイズ	：src_size で指定（2～255 バイト）
		情報データ	：エラー訂正を行うデータ（1～254 バイト）
		検査データ	：エラー訂正を行うためエンコードで付加された検査データ
		検査データサイズ	：check_size で指定（1～127 バイト）



情報データ及び検査データは、リードソロモン符号エンコードした結果を入力します。
エンコード結果は、原始多項式の0次の係数がLSBであることを想定しています。
参考情報として、ガロア体のべき表現とベクトル表現の対応は、α^0、α^1、α^2に対して、0x01、0x02、0x04となります。

	出力データ	アドレス	：dst で指定
		データサイズ	：src_size
		情報データ（訂正済み）	：エラー訂正済みの情報データ （サイズは情報データと同一）
		検査データ（訂正済み）	：エラー訂正済みの検査データ （サイズは検査データと同一）



	エラー訂正数	アドレス : correct_addr で指定 データサイズ : 1 バイト <内容> リードソロモン符号デコードの結果を出力します。デコードの結果は、エラーを訂正した数になります。入力データにエラーが無い場合は 0 を出力します。エラー訂正に失敗した場合は 0xff を出力します。
タイル数	1	
分割処理	不可	
解説	本機能は、src で指定された入力データに対して、下記の仕様に対応したリードソロモン符号デコードを行い、dst で指定されたアドレスに訂正済みのデータ、correct_addr で指定されたアドレスにエラー訂正数を出力します。 リードソロモン符号デコードの仕様 : ・ガロアフィールド : GF(2^8) ・ガロア体の原始多項式 : primitive で指定 ・シンボルあたりのビット数 : 8 ガロア体の原始多項式は、0 次の係数を LSB として設定します。 例えば、$\chi^8+\chi^4+\chi^3+\chi^2+1$ を設定する場合は、primitive に 0x11D を設定します。 リードソロモン符号デコードによるエラー訂正数は correct_addr で指定されたアドレスに格納されます。エラー訂正に成功した場合は訂正数、失敗した場合は 0xff が correct_addr で指定されたアドレスに格納されます。エラー訂正に失敗した場合は、訂正は行われず出力データは変化しません。 エラー訂正可能なシンボル数は、floor(check_size ÷ 2)となります。従って、check_size に 1 を設定した場合、訂正は行われず出力データは変化しません。エラー訂正数には 0 もしくは 0xff が出力されます。 本機能は、シンドローム算出、ユークリッド互除法、チェン探索、エラー値算出の順番でデコード処理を行います。エラーがない場合は、シンドローム算出で処理を停止します。エラーがある場合は、エラー値算出まで行いますが、エラーの数が多くなるにつれて、ユークリッド互除法とエラー値算出の処理時間が大きくなります。 従って、入力データのエラー数により処理時間が変動します。	
注意	入力データ内のエラー数が訂正可能なシンボル数より多い場合は、誤訂正が発生することがあります。誤訂正とは不正なデコード処理を行う事で、エラー訂正に失敗しているのに「エラー訂正数」が失敗（0xff）とならず、エラーではないシンボルを変更することもあります。	

4.13.3 Thinning

Thinning

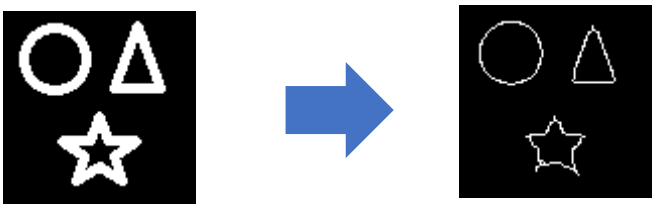
細線化した画像を出力します

コンフィグレーションデータファイル			r_drp_thinning.dat
対応バージョン			1.00
コンフィグレーションデータサイズ（バイト）			119456
ヘッダファイル			r_drp_thinning.h
パラメータ	構造体名		
	r_drp_thinning_t		
	メンバ名	型	説明
	src	uint32_t	入力データのアドレス
	dst	uint32_t	出力データのアドレス
	width	uint16_t	画像の幅（ピクセル）
	height	uint16_t	画像の高さ（ピクセル）
	result	uint32_t	処理結果のアドレス
	top	uint8_t	1:上端の境界処理あり 0:上端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の上端にあたる場合は 1 を指定してください。
	bottom	uint8_t	1:下端の境界処理あり 0:下端の境界処理なし 入力画像を分割しない場合は 1 を指定してください。 入力画像を分割して処理する場合、入力画像が元画像の下端にあたる場合は 1 を指定してください。
	step	uint8_t	繰り返し処理の種別 0:step1 1:step2 奇数回目の処理は step1 を指定してください。 偶数回目の処理は step2 を指定してください。 詳細は解説を参照してください。
	reverse	uint8_t	0：白部分を細線化します 1：黒部分を細線化します
	threshold	uint8_t	二値化の閾値（0～255）
入出力詳細	入力画像	アドレス	： src で指定
		幅（ピクセル）	： width で指定（128～1280、8 の整数倍）
		高さ（ピクセル）	： height で指定（16～960）
		フォーマット	： 8bit グレイスケール （1 ピクセルあたり 1 バイト） 0 は黒、1 以上は白として扱います。
		データサイズ	： (width) × (height) × 1 バイト
	出力画像	アドレス	： dst で指定
		幅（ピクセル）	： 入力画像と同じ
		高さ（ピクセル）	： 入力画像と同じ
		フォーマット	： 8bit グレイスケール（0 or 255） （1 ピクセルあたり 1 バイト） 黒を 0、白を 255 として出力します。
		データサイズ	： (width) × (height) × 1 バイト

	処理結果	アドレス	: result で指定 （src、dst と異なるアドレスにしてください）
		データサイズ	: 4 バイト
		フォーマット	: 白部分を黒に変更したピクセル数（4 バイト） （0～width×height）
		＜内容＞ 細線化の処理を行った結果として、白部分を黒（もしくは黒部分を白）に変更したピクセル数を格納する領域です。 白部分を黒（もしくは黒部分を白）に変更した個数が 0 であった場合、細線化の処理が終了したことを示します。 詳細は解説を参照してください。	
タイル数	3		
分割処理	可		

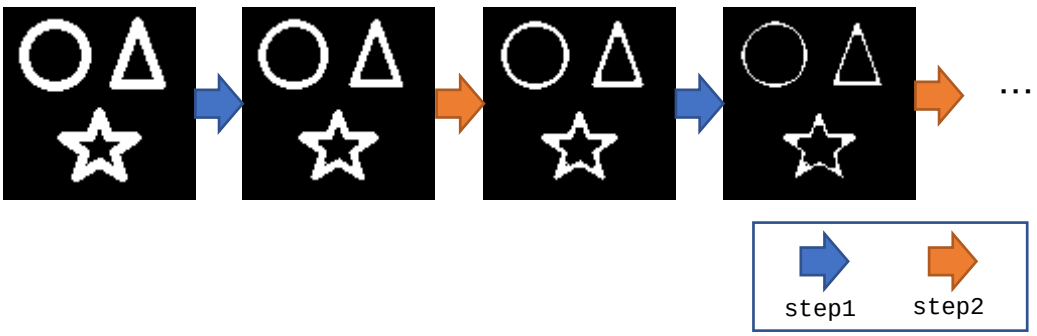
解説

本機能は、src で指定したアドレスの画像を二値化し、白部分（もしくは黒部分）の細線化を行い、dst で指定したアドレスに細線化の結果を出力します。result で指定したアドレスに白部分を黒（黒部分を白）に変更したピクセル数を出力します。以下の説明においては、reverse の設定が 0 の場合を想定します。reverse の設定が 1 の場合は「白部分を黒に変更する」を「黒部分を白に変更する」に読み替えて下さい。二値化は入力データが threshold を超えた場合は白、threshold 以下の場合は黒として扱います。



入力データ　　出力データ

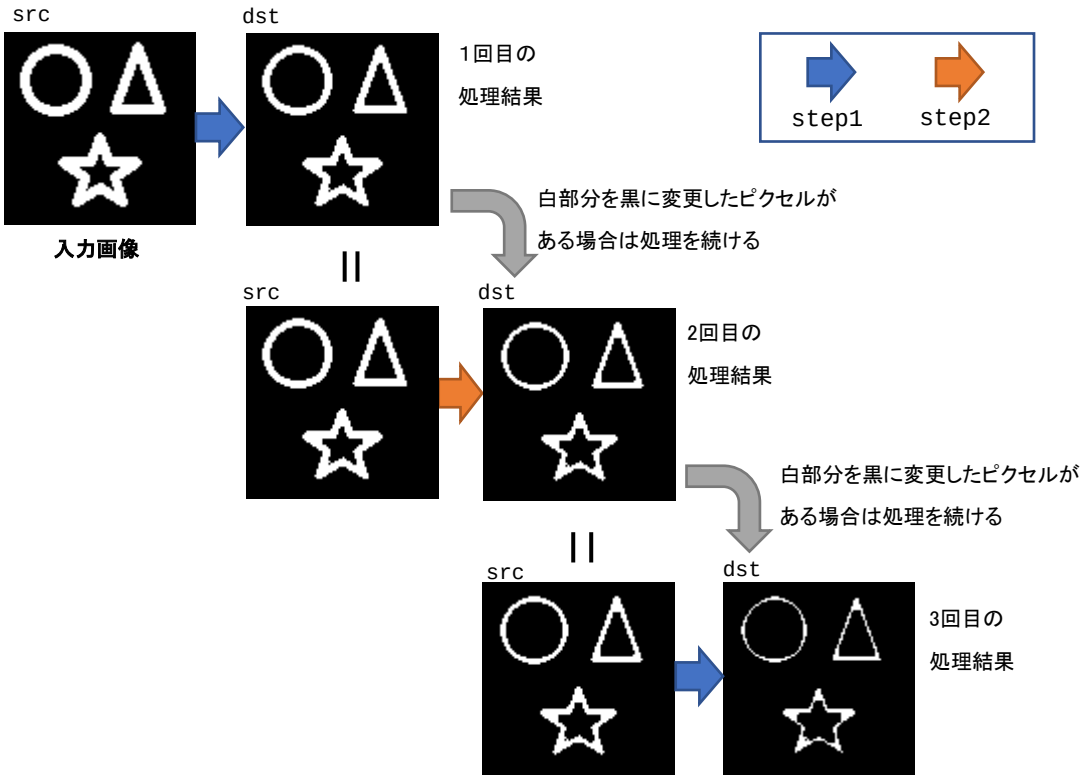
細線化を行う場合、白部分を黒に変更するアルゴリズムに基づき繰り返し処理を行う必要がありますが、本機能では Zhang-Suen のアルゴリズムを使用しています。Zhang-Suen のアルゴリズムは 2 種類（便宜上 step1、step2 と呼びます）の処理を交互に行います。
step1 と step2 それぞれに白部分を黒に変更する条件が存在し、条件は注目ピクセルを中心とした 3 × 3 ピクセルによって決められます。本機能では、境界処理において入力画像の範囲外のピクセルを黒として扱います。



step1 もしくは step2 の処理を行った時に、白部分を黒に変更したピクセルが存在する場合（result で指定した処理結果が 1 以上）は、出力画像に出力された結果を入力画像に設定し、前回の処理が step1 なら step2 を、step2 なら step1 の処理を引き続き行います。
step1 もしくは step2 の処理を行った時に、白部分を黒に変更したピクセルが存在しない場合（result で指定した処理結果が 0）は、細線化を終了します。

分割処理を行う場合は、分割された全ての処理で、白部分を黒に変更したピクセルが存在しなくなるまで、繰り返し処理を行います。

処理の最大時間を固定するために、白部分を黒に変更したピクセルが存在する場合でも細線化を終了することが可能です。ただし、出力画像は細線化が完了されていない途中結果になります。



本機能は、分割処理を行っていない場合、src と dst に同一アドレスを指定することが可能です。

注意	なし

4.13.4 ImageMerging

ImageMerging

分割して撮影された 2 枚のグレイスケール画像をマージします

コンフィグレーションデータファイル	r_drp_image_merging.dat
対応バージョン	1.00
コンフィグレーションデータサイズ（バイト）	851360
ヘッダファイル	r_drp_image_merging.h

パラメータ　構造体名

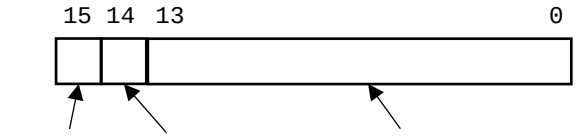
r_drp_image_merging_t

メンバ名	型	説明
src_adr1	uint32_t	入力画像 1 のアドレス
src_adr2	uint32_t	入力画像 2 のアドレス
dst_adr	uint32_t	出力画像のアドレス
width	uint16_t	画像の幅
height1	uint16_t	入力画像 1 の高さ
height2	uint16_t	入力画像 2 の高さ
search_window_w	uint8_t	テンプレート選定領域の幅（34～128、2 の整数倍） テンプレート選定領域の詳細は解説を参照してください。
search_window_h	uint8_t	テンプレート選定領域の高さ（34～128、2 の整数倍） テンプレート選定領域の詳細は解説を参照してください。
search_window1_x	uint16_t	テンプレート選定領域 1 の左上の X 座標 設定範囲の詳細は解説を参照してください。
search_window1_y	uint16_t	テンプレート選定領域 1 の左上の Y 座標 設定範囲の詳細は解説を参照してください。
search_window2_x	uint16_t	テンプレート選定領域 2 の左上の X 座標 設定範囲の詳細は解説を参照してください。
search_window2_y	uint16_t	テンプレート選定領域 2 の左上の Y 座標 設定範囲の詳細は解説を参照してください。
search_window3_x	uint16_t	テンプレート選定領域 3 の左上の X 座標 設定範囲の詳細は解説を参照してください。
search_window3_y	uint16_t	テンプレート選定領域 3 の左上の Y 座標 設定範囲の詳細は解説を参照してください。
search_window4_x	uint16_t	テンプレート選定領域 4 の左上の X 座標 設定範囲の詳細は解説を参照してください。
search_window4_y	uint16_t	テンプレート選定領域 4 の左上の Y 座標 設定範囲の詳細は解説を参照してください。
src_diff	uint16_t	入力画像 1 と入力画像 2 の Overlap 領域の高さを決める為の値（0 ～ search_window*_y※ - max_ga_y - 2） ※ テンプレート選定領域 1～4 の左上の Y 座標の中で最小値 Overlap 領域の定義は解説を参照してください。
max_gap_x	uint8_t	テンプレートとマッチング対象の X 方向のズレの最大値（0～126） 処理速度やマッチング誤判定防止の観点から 16 以下を推奨値としています。
max_gap_y	uint8_t	テンプレートとマッチング対象の Y 方向のズレの最大値（0～126） 処理速度やマッチング誤判定防止の観点から 16 以下を推奨値としています。
angledata_adr	uint32_t	回転補正時の角度データのアドレス
angle_times	uint8_t	回転補正回数（0～255） 0 を指定した場合、回転補正は行いません。
result_adr	uint32_t	マージ情報のアドレス
padding_value	uint8_t	参照する入力画像が範囲外となったときの出力値

入出力詳細	入力画像 1	アドレス	: src_adr1 で指定
		幅（ピクセル）	: width で指定（最大値 1920、2 の整数倍）
		高さ（ピクセル）	: height1 で指定（最大値 1080、2 の整数倍）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height1) × 1 バイト
	入力画像 2	アドレス	: src_adr2 で指定
		幅（ピクセル）	: 入力画像 1 と同じ
		高さ（ピクセル）	: height2 で指定（最大値 1080、2 の整数倍）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height2) × 1 バイト
	出力画像	アドレス	: dst_adr で指定
		幅（ピクセル）	: 入力画像 1 と同じ
		高さ（ピクセル）	: height2 + src_diff + max_gap_y（最大値）
		フォーマット	: 8bit グレイスケール（1 ピクセルあたり 1 バイト）
		データサイズ	: (width) × (height2 + src_diff + max_gap_y) × 1 バイト
	角度データ（入力）	アドレス	: angledata_adr で指定
		データサイズ	: angle_times ×4 バイト
		フォーマット	: アドレス先頭から、 回転角度に対する sin 値（2 バイト）、 回転角度に対する cos 値（2 バイト）、

＜内容＞
アドレスの先頭から、回転角度に対する sin 値と cos 値を 4 バイト単位で回転補正回数だけ入力します。
本機能は、入力画像が回転していないことを前提としていますが、角度データを設定することで微小の回転を補正することができます。angledata_adr に 0 を設定した場合は、回転補正を行いません。angledata_adr に 0 以外を設定した場合は、設定された角度データで回転補正を行います。設定した角度データに加えて、回転補正を行わないでマッチングしたい場合は、0 度の角度データを設定してください。
回転角度を大きく設定すると、テンプレートとマッチング判定領域が重ならず、画像をきれいにつなげられない可能性があります。また、出力画像の品質の点からも回転角度は、±5 度程度までを推奨とします。

sin 値及び cos 値のフォーマットは下図の通りです。



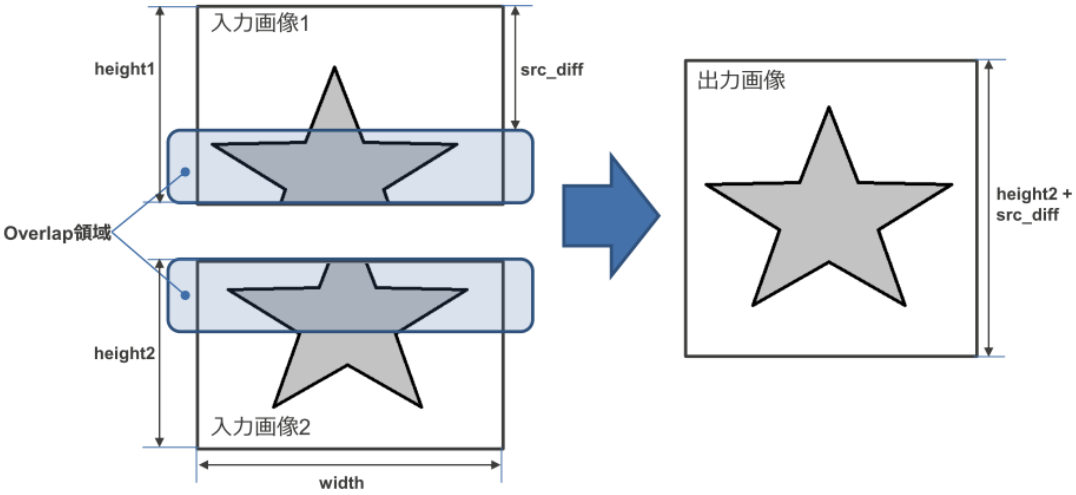
2の補数表現

詳細は解説を参照してください。

マージ情報	アドレス	:	result_adr で指定
	データサイズ	:	36 バイト
	フォーマット	:	アドレス先頭から、 出力画像の幅（2 バイト）、 出力画像の高さ（2 バイト）、 マッチング結果の X 方向ズレ量（2 バイト）、 マッチング結果の Y 方向ズレ量（2 バイト）、 マッチング結果の回転角度に対する sin 値（2 バイト）、 マッチング結果の回転角度に対する cos 値（2 バイト）、 選定したテンプレート 1 の左上の X 座標値（2 バイト）、 選定したテンプレート 1 の左上の Y 座標値（2 バイト）、 選定したテンプレート 2 の左上の X 座標値（2 バイト）、 選定したテンプレート 2 の左上の Y 座標値（2 バイト）、 選定したテンプレート 3 の左上の X 座標値（2 バイト）、 選定したテンプレート 3 の左上の Y 座標値（2 バイト）、 選定したテンプレート 4 の左上の X 座標値（2 バイト）、 選定したテンプレート 4 の左上の Y 座標値（2 バイト）、 テンプレート 1 のマッチング結果の X 方向ズレ量 （1 バイト）、 テンプレート 1 のマッチング結果の Y 方向ズレ量 （1 バイト）、 テンプレート 2 のマッチング結果の X 方向ズレ量 （1 バイト）、 テンプレート 2 のマッチング結果の Y 方向ズレ量 （1 バイト）、 テンプレート 3 のマッチング結果の X 方向ズレ量 （1 バイト）、 テンプレート 3 のマッチング結果の Y 方向ズレ量 （1 バイト）、 テンプレート 4 のマッチング結果の X 方向ズレ量 （1 バイト）、 テンプレート 4 のマッチング結果の Y 方向ズレ量 （1 バイト） ＜内容＞ マージ結果の情報を出力します。 sin 値及び cos 値のフォーマットは角度データと同じです。 また、回転補正を行わない場合、sin 値=0x0000、cos 値=0x4000 が出力されます。
タイル数	6		
分割処理	不可		

解説

本機能は、src_adr1 と src_adr2 で指定したアドレスに格納されている 2 枚のグレイスケール画像をマージし、dst_adr で指定したアドレスに出力します。2 枚の画像の縮尺は同じである必要があり、縦方向にマージされます。



本機能は、入力画像 1 と入力画像 2 の重なり（Overlap 領域）のおおよその値を、予めユーザーが把握している必要があります。この領域内で入力画像 1 と入力画像 2 の画像にズレがあっても、微小であれば補正してマージすることができます。そのズレの最大値は max_gap_x と max_gap_y で指定します。また、微小であれば画像が回転していても補正してマージすることができます。その回転量は angledata_adr と angle_times で指定します。

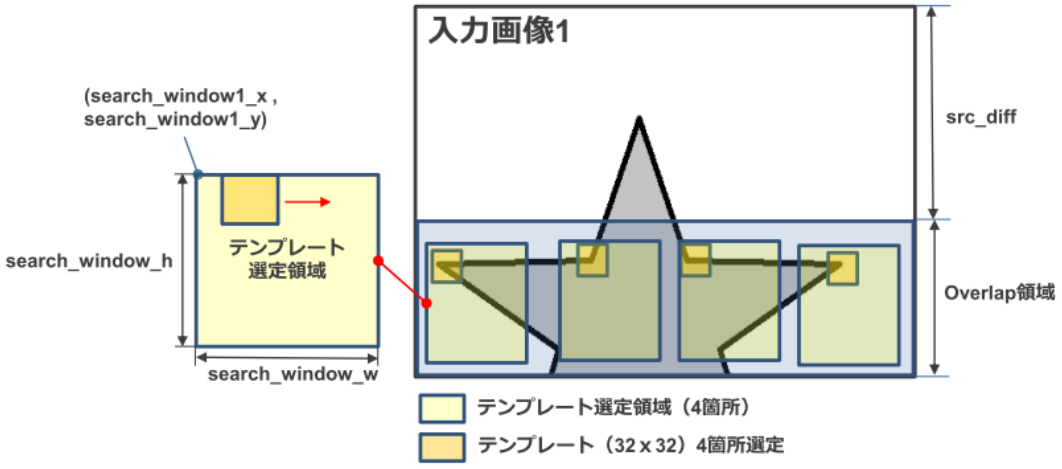
本機能は、以下のように処理を行います。

1. テンプレート選定
2. テンプレート 1～4 のマッチング
3. 画像全体のマッチング
4. 画像のマージ

各処理の詳細を次頁以降で説明します。

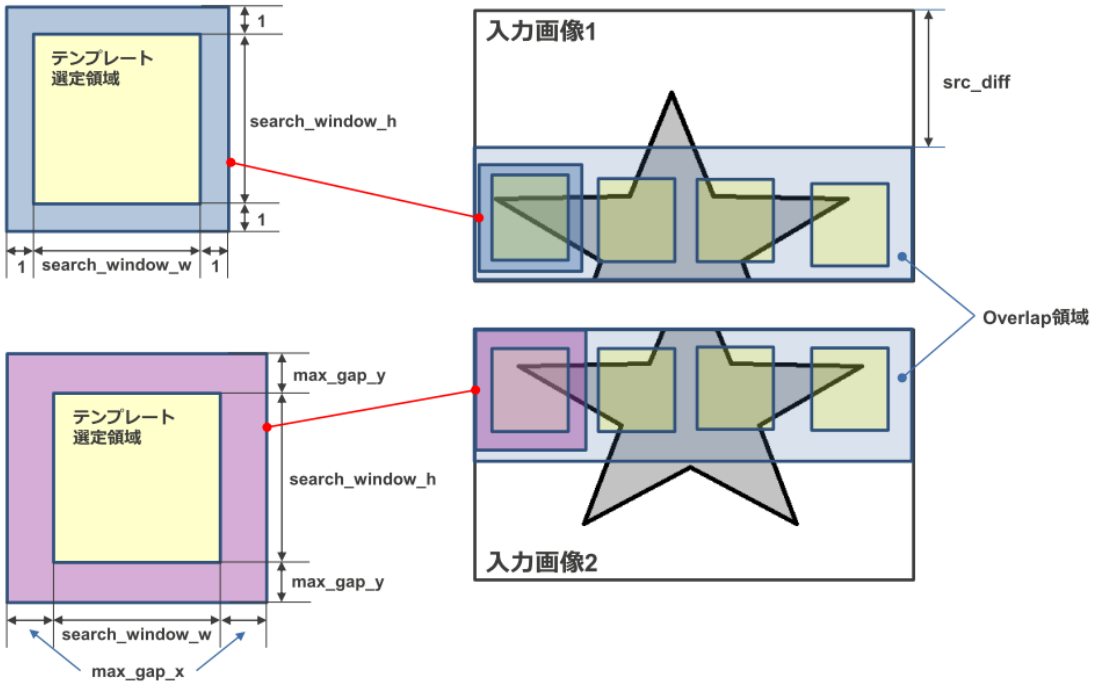
1. テンプレート選定

本機能は、2 つの入力画像を特徴箇所（テンプレート）でマッチングして画像をマージします。本機能では入力画像 1 から 4 つのテンプレートを選定します。この選定のため、入力画像 1 の Overlap 領域に 4 つのテンプレート選定領域を指定します。指定されたテンプレート選定領域内でエントロピー（3×3 の sobel フィルタの合計値）が最大になる 32×32 ピクセルの領域をテンプレートとしています。



テンプレート選定領域は、以下の 2 つの条件を満たすように設定してください。

（ア） テンプレート選定領域は、Overlap 領域の 1 ピクセル以上、内側に設定してください
（イ） テンプレート選定領域は、Overlap 領域の max_gap_x、max_gap_y 以上、内側に設定してください



Sobel フィルタについては、4.10.4 Sobel を参照してください。

2. テンプレート 1～4 のマッチング

入力画像 1 上の選定されたテンプレートが、入力画像 2 上のどの位置に当たるかを調べます。

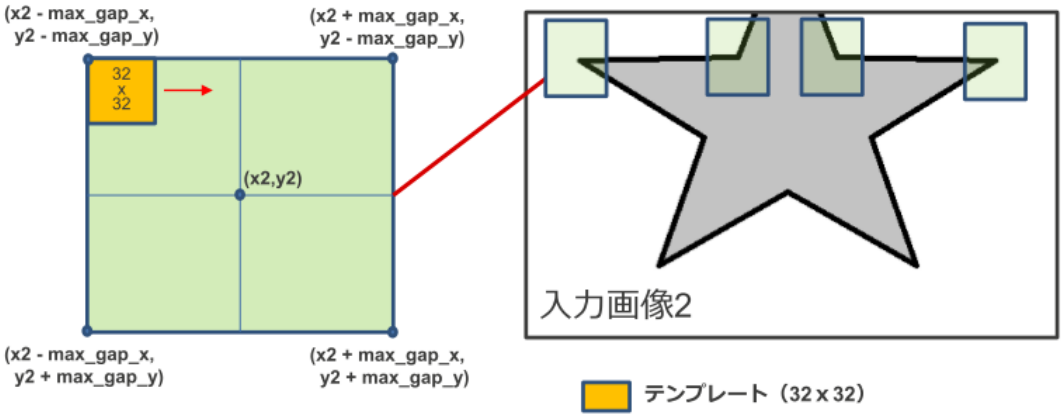
入力画像 1 上のテンプレートの左上座標を(x1, y1)とすると、入力画像 2 における対応座標(x2,y2)は以下になります。

```
(x2,y2)= (x1, y1 - src_diff)
```

(x2,y2)を起点とする 32 x 32 の領域と、入力画像 1 のテンプレートに対し、SAD 法によるマッチング処理を行います。SAD の値は、入力画像 2 の輝度値を I(x,y)、テンプレート上の輝度値を T(x,y)とした場合、以下の計算により求まります。

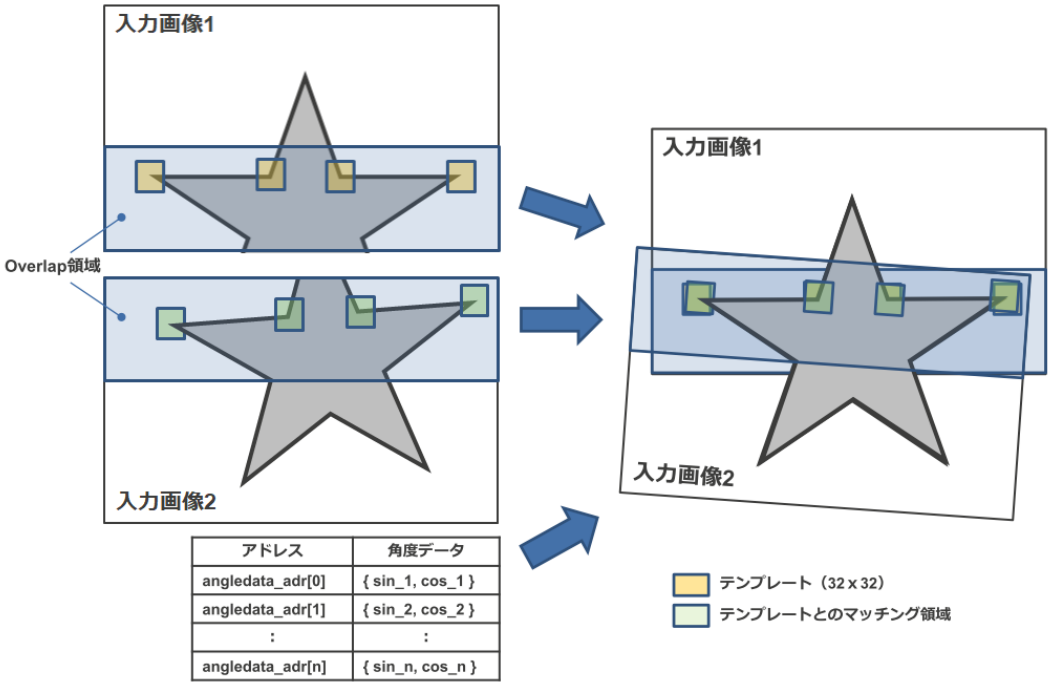
$$SAD = \sum_{y}^{y+32} \sum_{x}^{x+32} |I(x,y) - T(x,y)|$$

(x2 - max_gap_x,y2 - max_gap_y)～(x2 + max_gap_x,y2 + max_gap_y)の範囲全ての SAD の値を計算して、テンプレート毎に SAD の値が最小値となる(x3,y3)を求め、テンプレート毎に X 方向ズレ量と Y 方向ズレ量を計算します。



3. 画像全体のマッチング

テンプレート 1～4 のマッチングで求めた X 方向ズレ量、Y 方向ズレ量と、回転補正の角度データを用いて、画像全体のマッチングを行います。

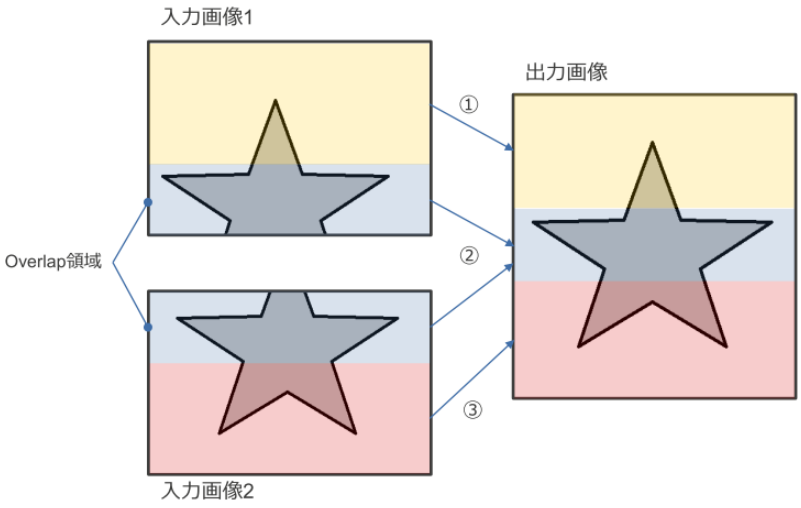


アドレス	角度データ
angledata_adr[0]	{ sin_1, cos_1 }
angledata_adr[1]	{ sin_2, cos_2 }
:	:
angledata_adr[n]	{ sin_n, cos_n }

4. 画像のマージ

入力画像 1 と入力画像 2 をマージし、マージ結果を出力画像領域に出力します。

① 出力画像領域の 1 行目から src_diff 行目までについて、入力画像 1 を出力画像領域にコピーします。
② 出力画像領域の src_diff+1 行目から height1 行目までについて、入力画像 1 と入力画像 2 を使用したマージ結果を出力します。
③ 出力画像領域の height1+1 行目からは、入力画像 2 を使用した結果を出力します。



本機能は、src_adr1 と dst_adr に同一アドレスを指定することが可能です。

注意	なし

5.　DRP Library 使用方法

本ライブラリを使用する場合、DRPの初期化、コンフィグレーションデータのロードなどが必要です。また、コンフィグレーションデータごとにパラメータが異なるため、使用するコンフィグレーションデータの仕様をもとにパラメータの設定を行ってください。

実際に DRP Library を使用したサンプルプログラムの詳細は、「RZ/A2M グループ 2D Barcode アプリケーションノート（R01AN4503）」を参照してください。

6. 関連ドキュメント

ユーザーズマニュアル：ハードウェア

RZ/A2M グループ ユーザーズマニュアル ハードウェア編（R01UH0746）
（最新版をルネサスエレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：ソフトウェア

RZ/A2M グループ DRP Driver ユーザーズマニュアル（R01US0355）
（最新版をルネサスエレクトロニクスホームページから入手してください。）

RZ/A2M グループ 2D Barcode アプリケーションノート（R01AN4503）
（最新版をルネサスエレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境

ルネサスエレクトロニクス統合開発環境（e2 studio）に関しては、最新版をルネサスエレクトロニクスホームページから入手してください。

テクニカルアップデート／テクニカルニュース

（最新の情報を入手するにはルネサス エレクトロニクスにお問い合わせください。）

改訂記録	RZ/A2M グループ DRP Library ユーザーズマニュアル

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2018.09.28	ー	新規発行
1.01	2018.12.28	6	表 1.1　DRP Library の機能一覧に以下の機能追加 ・Prewitt ・Opening ・Closing ・ResizeBilinearFixed ・ResizeNearest ・CircleFitting ・Histogram
		7	2 動作条件　RENESAS e2 studio のバージョンを 7.3.0 に変更
		8,9	3 ファイル構成　コンフィグレーションデータ、ヘッダファイルを追加
		10	4.1 DRP Library 仕様の読み方　分割処理の説明を追加
		11	4.2 Simple ISP 章追加
		16	4.3.1 BinarizationFixed　解説欄の参考 URL を変更
		22	4.3.4 Dilate パラメータ欄 top,bottom の説明を変更 解説欄の参考 URL を変更、説明を追記
		24	4.3.5 Erode パラメータ欄 top,bottom の説明を変更 解説欄の参考 URL を変更、説明を追記
		27	4.3.7 GaussianBlur パラメータ欄 top,bottom の説明を変更 解説欄の参考 URL を変更、説明を変更
		28	4.3.8 MediaBlur パラメータ欄 top,bottom の説明を変更 解説欄の参考 URL を変更、説明を変更
		29	4.3.9 Sobel パラメータ欄 top,bottom の説明を変更 解説欄の説明を変更
		31	4.3.10 Prewitt　章追加
		35	4.3.12 UnsharpMasking パラメータ欄 top,bottom の説明を変更 解説欄の参考 URL を変更、説明を変更
		37	4.3.13 Opening　章追加
		40	4.3.14 Closing　章追加
		44	4.4.2 Bayer2Grayscale パラメータ欄 top,bottom の説明を変更 解説欄の参考 URL を変更、説明を変更
		56	4.4.6 ResizeBilinearFixed タイトルを ResizeBilinear（バイリニア補間）から ResizeBilinearFixed（バイリニア法・固定倍率）に変更 入出力詳細 入力画像の幅、データサイズの記載を修正 解説欄の参考 URL を変更

Rev.	発行日	改訂内容	
		ページ	ポイント
1.01	2018.12.28	57	4.4.7 ResizeBilinear　章追加
		59	4.4.8 ResizeNearest　章追加
		63	4.5.1 CannyCalculate パラメータ欄 top,bottom の説明を変更 解説欄の参考 URL を変更
		67	4.5.3 CornerHarris 解説欄の図、参考 URL、説明を変更
		69	4.5.4 CircleFitting　章追加
		96	4.6.3 Histogram　章追加
1.02	2019.04.15	6	表 1.1 DRP Library の機能一覧に以下の機能追加 ・Laplacian ・Bayer2Rgb ・ImageRotate ・Affine ・MinutiaeExtract ・MinutiaeDelete ・Thinning ・ReedSolomonGf8
		8,9	3 ファイル構成　コンフィグレーションデータ、ヘッダファイルを追加
		12,13	4.2 Simple ISP 図 4.1 Simple ISP ブロック図を変更 バージョン変更に伴いバージョン、コンフィグレーションデータサイズを変更 パラメータ欄 width の説明を変更
		34	4.3.11 Laplacian　章追加
		47	4.4.3 Bayer2Rgb　章追加
		54	4.4.5 ImageRotate　章追加
		61	4.4.9 Affine　章追加
		74	4.5.5 MinutiaeExtract　章追加
		81	4.5.6 MinutiaeDelete　章追加
		91	4.5.7 Thinning　章追加
		95	4.6.1 ReedSolomon ・入力データサイズ src_size の最大値変更 ・入出力詳細、解説、注意の記載変更
		97	4.6.2 ReedSolomonGf8　章追加
1.03	2019.05.31	40	4.3.13 HistogramNormalization　章追加
		43	4.3.14 HistogramNormalizationRgb　章追加
		62	4.4.4 Bayer2RgbColorCorrection　章追加
		66	4.4.6 CroppingRgb　章追加
		71	4.4.9 ResizeBilinearFixedRgb　章追加
		88	4.5.5 FindContours　章追加
1.04	2019.09.30	12	4.1 DRP Library 仕様の読み方 パラメータの説明を変更
		17	4.2.3 Simple Isp API YCbCr422 の出力フォーマットを追加

		40	4.3.13 HistogramNormalization 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		43	4.3.14 HistogramNormalizationRgb 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		62	4.4.4 Bayer2RgbColorCorrection 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		66	4.4.6 CroppingRgb 対応バージョンを 1.00 に更新
		71	4.4.9 ResizeBilinearFixedRgb 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		78	4.5.1 CannyCalculate 入力画像の高さの範囲を変更 解説の誤記を修正
		88	4.5.5 FindContours 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更 入力画像の幅、高さの最小値を変更 矩形情報の最大出力個数の最小値を変更
1.05	2019.12.17	9,10	3. ファイル構成を変更
		14	4.2.1 Simple ISP 概要 図 4.2 を追加
		16,18	4.2.3 Simple Isp API パラメータの説明を変更
		30	4.3.6 GammaCorrection 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更 パラメータ欄　table を追加 解説欄を変更
		42	4.3.13 HistogramNormalization 解説の src_pixel_rstd の計算式を修正
		46	4.3.14 HistogramNormalizationRgb 解説の src_pixel_*_rstd の計算式を修正
1.06	2020.03.31	15	4.2.3 Simple Isp API 対応バージョンを 1.01 に更新 コンフィグレーションデータサイズを変更
1.07	2020.06.30	7,8	表 1.1、表 1.2　DRP Library の機能一覧に以下の機能追加 ・Simple ISP with object detection by color (HSV) ・Simple ISP with background subtraction ・Simple ISP with object detection using sobel ・Simple ISP with distortion correction ・Simple ISP with scaling and normalization(32bit) ・Simple ISP with color calibration and 3DNR ・Remap ・ImageMerging DRP Library のカテゴリ分けを変更

		9	2 動作条件　RENESAS e2 studio のバージョンを 7.8.0 に変更
		10,11	3 ファイル構成　コンフィグレーションデータ、ヘッダファイルを追加
		14	4.2.1 Simple ISP 概要　説明を変更
		15	4.2.2 Simple ISP ライブラリ構成 YCbCr Planar format, RGB の出力フォーマットに対応
		16,17,18 ,20	4.2.3 Simple Isp API YCbCr Planar format, RGB の出力フォーマットに対応 表 4.2 を追加
		22	4.3 Simple ISP with object detection by color (HSV)　章追加
		28	4.4 Simple ISP with background subtraction　章追加
		35	4.5 Simple ISP with object detection using sobel　章追加
		40	4.6 Simple ISP with distortion correction　章追加
		46	4.7 Simple ISP with scaling and normalization (32bit)　章追加
		51	4.8 Simple ISP with color calibration and 3DNR　章追加
		-	4.9、4.10、4.11、4.12、4.13　DRP Library のカテゴリ分け変更
		60	4.9.1 Bayer2Grayscale 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		62	4.9.2 Bayer2Rgb 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		68	4.9.3 Bayer2RgbColorCorrection 対応バージョンを 1.01 に更新 コンフィグレーションデータサイズを変更
		71	4.9.4 Argb2Grayscale 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		72	4.9.5 BinarizationFixed 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		73,75	4.9.6 BinarizationAdaptive 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更 解説欄の注目ピクセルを含むブロックが、入力画像の左端 2 ブロック以内の場合の記載を変更
		76	4.9.7 BinarizationAdaptiveBit 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		78	4.9.8 GammaCorrection 対応バージョンを 1.01 に更新 コンフィグレーションデータサイズを変更
		79	4.9.9 Cropping 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		80	4.9.10 CroppimgRgb 対応バージョンを 1.01 に更新 コンフィグレーションデータサイズを変更

		81	4.9.11 ResizeBilinearFixed 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		82	4.9.12 ResizeBilinearFixedRgb 対応バージョンを 1.01 に更新 コンフィグレーションデータサイズを変更
		83	4.9.13 ResizeBilinear 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		85	4.9.14 ResizeNearest 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		86	4.9.15 ImageRotate 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更 出力画像の幅、データサイズの記載を変更
		89	4.9.16 Affine 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		92	4.9.17 Remap　章追加
		97	4.10.1 MedianBlur 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		98	4.10.2 GaussianBlur 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		99	4.10.3 UnsharpMasking 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		101	4.10.4 Sobel 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		103	4.10.5 Prewitt 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		105	4.10.6 Laplacian 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		107	4.10.7 Dilate 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		109	4.10.8 Erode 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		117	4.11.1 CannyCalculate 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更

		119	4.11.2 CannyHysterisis 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		121	4.11.3 CornerHarris 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		123	4.11.4 MinutiaeExtract 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		130	4.11.5 MinutiaeDelete 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		140	4.11.6 CircleFitting 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		144	4.11.7 FindContours 対応バージョンを 1.01 に更新 コンフィグレーションデータサイズを変更
		148	4.12.1 Histogram 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		154	4.12.2 HistogramNormalization 対応バージョンを 1.01 に更新 コンフィグレーションデータサイズを変更
		157	4.12.3 HistogramNormalizationRgb 対応バージョンを 1.01 に更新 コンフィグレーションデータサイズを変更
		161	4.13.1 ReedSolomon 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		163	4.13.2 ReedSolomonGf8 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		165	4.13.3 Thinning 対応バージョンを 1.00 に更新 コンフィグレーションデータサイズを変更
		169	4.13.4 ImageMerging　章追加
1.08	2020.09.30	5	1.2 機能 Simple ISP のカテゴリ名を Image processing に変更
		19	4.2.3 Simple ISP API 平均輝度の計算式を変更
1.09	2020.12.25	168	4.13.4 ImageMerging src_diff の説明を変更

RZ/A2M グループ DRP Library ユーザーズマニュアル

発行年月日　2018年9月28日　Rev.1.00
2020年12月25日　Rev.1.09

発行　ルネサス エレクトロニクス株式会社
〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

RZ/A2M グループ

ルネサス エレクトロニクス株式会社

R01US0367JJ0109

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/

Colophon 3.1