cv_lite API Reference Manual

> This manual is sourced from the Canaan Technology official website and is backed up here for your convenience. Due to version iterations and other factors, the APIs in this manual are not guaranteed to be fully functional. All API methods that have been tested and confirmed to be functional will be presented as tutorials in subsequent sections of this chapter.

API Introduction

grayscale_find_blobs

Description

Finds blobs (connected regions) in a grayscale image and returns their locations.

Syntax

Please ensure that the sensor is configured to output grayscale images, otherwise an error will occur.

```
import cv_lite

image_shape = [480,640]  # Height, width

threshold = [230, 255]   # Binarization threshold range (bright area)
min_area = 10            # Minimum target area
kernel_size = 1          # Corrosion kernel size (can be used for noise
reduction)

img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get image data reference

# Perform binary connected domain detection on grayscale images
blobs = cv_lite.grayscale_find_blobs(image_shape, img_np,threshold[0],
threshold[1],min_area, kernel_size)
```

Parameter

Parameter Name	Description	Input/Output	Note
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
threshold_min	Minimum value of the binarization threshold, int type	Input	
threshold_max	Maximum value of the binarization threshold, int type	Input	
min_area	Minimum area of the region, int type	Input	
kernel_size	Kernel size, int type	Input	

Return Value

Return Value	Description
blobs	List of blob location information, each containing 4 elements representing the location of a blob, including x, y, w, and h.

rgb888_find_blobs

Description

Finds blobs (connected regions) in an RGB888 image and returns their locations.

Syntax

Please ensure that the sensor is configured to output an RGB888 image; otherwise, an error will occur.

```
import cv_lite

image_shape = [480,640]  # Height, width

threshold = [120, 255, 0, 50, 0, 50]

min_area = 100     # Minimum color block area
kernel_size = 1    # Corrosion dilation kernel size (for preprocessing)

# Capture a frame of image
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get a reference to an RGB888 ndarray

# Call the cv_lite extension to perform color block detection and return a list of [x, y, w, h, ...]
blobs = cv_lite.rgb888_find_blobs(image_shape, img_np, threshold, min_area, kernel_size)
```

Parameters

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
threshold	Binarization threshold range, list type, including the threshold ranges for the R, G, and B channels	Input	
min_area	Minimum region area, int type	Input	
kernel_size	Kernel size, int type	Input	

Return Value

Return Value	Description
blobs	Blob location information list, each containing 4 elements representing the location of a blob, including the x, y, w, and h positions

grayscale_find_circles

Description

Finds a circle in a grayscale image and returns its location.

Syntax

Please ensure that the sensor is configured to output a grayscale image; otherwise, an error will occur.

```
import cv_lite

image_shape = [480,640]  # Height, width

# -------------------------------
# Hough circle detection parameters
# -------------------------------
dp = 1            # Inverse ratio of resolution
minDist = 20      #  Minimum distance between centers
param1 = 80       #  Upper threshold for Canny edge detector
param2 = 20       #  Accumulator threshold for center detection
minRadius = 10    #  Minimum radius to detect
maxRadius = 50    #  Maximum radius to detect

#  Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  #  Get image data reference

# Detect circles using Hough Transform
# Return format: [x1, y1, r1, x2, y2, r2, ...]
circles = cv_lite.grayscale_find_circles(image_shape, img_np,dp, minDist,param1,
param2,
minRadius, maxRadius)
```

Parameters

Parameter Name	Description	Input/Output	Explanation
image_shape	Image shape, list type, in the order [height, width], e.g., [480, 640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
dp	Accumulator resolution ratio, float type	Input	
minDist	Minimum distance from circle center, int type	Input	
param1	Canny high threshold, int type	Input	
param2	Accumulator threshold, int type	Input	
minRadius	Minimum circle radius, int type	Input	
maxRadius	Maximum circle radius, int type	Input	

Return Value

Return Value	Description
circles	A list of circle information. Each 3 elements represent the information of a circle, including the x, y, and r positions.

rgb888_find_circles

Description

Finds a circle in an RGB888 image and returns its location.

Syntax

Please ensure that the sensor is configured to output an RGB888 image; otherwise, an error will occur.

```
import cv_lite

image_shape = [480,640]  # Height, width

# -------------------------------
#  Hough Circle parameters
# -------------------------------
dp = 1           # Inverse ratio of accumulator resolution
minDist = 30     # Minimum distance between detected centers
param1 = 80      # Higher threshold for Canny edge detection
param2 = 20      # Threshold for center detection in accumulator
minRadius = 10   # Minimum circle radius
maxRadius = 50   # Maximum circle radius

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get a reference to an RGB888 ndarray

# Call the Hough circle detection function extended by cv_lite and return the circle parameter list [x, y, r, ...]
circles = cv_lite.rgb888_find_circles(image_shape, img_np, dp, minDist, param1, param2, minRadius, maxRadius)
```

Parameters

Parameter Name	Description	Input/Output	Explanation
image_shape	Image shape, list type, in the order [height, width], e.g., [480, 640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
dp	Inverse ratio of accumulator resolution to image resolution, float type	Input	
minDist	Minimum distance to the detected circle center, int type	Input	
param1	Canny edge detection high threshold, int type	Input	
param2	Hough transform circle center detection threshold, int type	Input	
minRadius	Minimum radius of the detected circle, int type	Input	
maxRadius	Maximum radius of the detected circle, int type	Input	

Return Value

Return Value	Description
circles	A list of circle information. Each 3 elements represents a circle, including its x, y, and r positions.

grayscale_find_rectangles

Description

Finds a rectangle in a grayscale image and returns its location.

Syntax

Please ensure that the sensor is configured to output grayscale images, otherwise an error will occur.

```
import cv_lite

image_shape = [480,640]  # Height, width

# -------------------------------
# Adjustable rectangle detection parameters
# -------------------------------
canny_thresh1      = 50        # Canny low threshold
canny_thresh2      = 150       # Canny high threshold
approx_epsilon     = 0.04      # Polygon approximation accuracy
area_min_ratio     = 0.001     # Min area ratio
```

```
max_angle_cos      = 0.3       # Max cosine of angle between edges
gaussian_blur_size = 5         # Gaussian blur kernel size

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()

# Call the underlying rectangle detection function
# Return format:[x0, y0, w0, h0, x1, y1, w1, h1, ...]
rects = cv_lite.grayscale_find_rectangles(
    image_shape, img_np,
    canny_thresh1, canny_thresh2,
    approx_epsilon,
    area_min_ratio,
    max_angle_cos,
    gaussian_blur_size
)
```

Parameter

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, in the order [height, width], e.g., [480, 640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
canny_thresh1	Canny edge detection lower threshold, int type	Input	
canny_thresh2	Canny edge detection upper threshold, int type	Input	
approx_epsilon	Polygon fitting accuracy ratio, float type	Input	
area_min_ratio	Minimum area ratio, float type	Input	
max_angle_cos	Maximum angle cosine, float type	Input	
gaussian_blur_size	Gaussian blur kernel size, int type	Input	

Return Value

Return Value	Description
rects	A list of rectangle position information, each containing 4 elements representing the position of a rectangle, including x, y, w, and h.

rgb888_find_rectangles

Description

Finds a rectangle in an RGB888 image and returns its location.

Syntax

Please ensure that the sensor is configured to output an RGB888 image; otherwise, an error will occur.

```
import cv_lite

image_shape = [480,640]  # Height, width

# -------------------------------
# Adjustable parameters (recommended for tuning)
# -------------------------------
canny_thresh1       = 50        # Canny edge low threshold
canny_thresh2       = 150       # Canny edge high threshold
approx_epsilon      = 0.04      # Polygon approximation precision (ratio)
area_min_ratio      = 0.001     # Minimum area ratio (0~1)
max_angle_cos       = 0.5       # Max cosine of angle (smaller closer to rectangle)
gaussian_blur_size  = 5         # Gaussian blur kernel size (odd number)

# Capture current frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get RGB888 ndarray reference

# Call the underlying rectangle detection function and return a rectangle list [x0, y0, w0, h0, x1, y1, w1, h1, ...]
# Call underlying rectangle detection function, returns list of rectangles [x, y, w, h, ...]
rects = cv_lite.rgb888_find_rectangles(
    image_shape, img_np,
    canny_thresh1, canny_thresh2,
    approx_epsilon,
    area_min_ratio,
    max_angle_cos,
    gaussian_blur_size
)
```

Parameter

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, in the order [height, width], e.g., [480, 640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
canny_thresh1	Canny edge detection lower threshold, int type	Input	
canny_thresh2	Canny edge detection upper threshold, int type	Input	
approx_epsilon	Polygon fitting accuracy ratio, float type	Input	
area_min_ratio	Minimum area ratio, float type	Input	
max_angle_cos	Maximum angle cosine, float type	Input	
gaussian_blur_size	Gaussian blur kernel size, int type	Input	

Return Value

Return Value	Description
rects	A list of rectangle position information, each containing 4 elements representing the position of a rectangle, including x, y, w, and h.

grayscale_find_edges

Description

Find edges in a grayscale image and return the edge-detected image as a ulab.numpy.ndarray. You can create an Image instance based on the returned data.

Syntax

Please ensure that the output image in the Sensor configuration is grayscale; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640]  # Height, width

# -------------------------------
# Canny edge detection thresholds
# -------------------------------
threshold1 = 50  # Lower threshold
threshold2 = 80  # Upper threshold

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get ndarray reference
```

```
# Call the cv_lite extension to perform edge detection
# Returns edge image ndarray
edge_np = cv_lite.grayscale_find_edges(
    image_shape, img_np, threshold1, threshold2)

# Wrap ndarray as image for display
img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
                        alloc=image.ALLOC_REF, data=edge_np)
```

Parameters

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, in the order [height, width], e.g., [480, 640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
threshold1	Low threshold, int type	Input	
threshold2	High threshold, int type	Input	

Return Value

Return Value	Description
edge_np	Image data after edge detection, ulab.numpy.ndarray type, used to create an image instance

rgb888_find_edges

Description

Find edges in an RGB888 image and return the edge-detected image data as a ulab.numpy.ndarray. You can then create an Image instance from this data.

Syntax

Please ensure that the output image in the Sensor configuration is RGB888; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640]  # Height, width

# -------------------------------
# Canny edge detection thresholds
# -------------------------------
threshold1 = 50  # Lower threshold
threshold2 = 80  # Upper threshold

# Capture a frame
```

```
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get a reference to an RGB888 ndarray

# Call the edge detection function of the cv_lite extension and return the
grayscale edge map ndarray
edge_np = cv_lite.rgb888_find_edges(image_shape, img_np, threshold1, threshold2)

# Construct a grayscale image object for display
img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
alloc=image.ALLOC_REF, data=edge_np)
```

Parameters

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, in the order [height, width], e.g., [480, 640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
threshold1	Low threshold, int type	Input	
threshold2	High threshold, int type	Input	

Return Value

Return Value	Description
edge_np	Image data after edge detection, ulab.numpy.ndarray type, used to create an image instance

grayscale_threshold_binary

Description

Binarizes a grayscale image and returns the binarized image data as a ulab.numpy.ndarray. This data can be used to create an Image instance.

Syntax

Please ensure that the output image of the Sensor configuration is grayscale; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640]  # Height, width

# -------------------------------
# Binary threshold parameters
# -------------------------------
thresh = 130   # Threshold value
maxval = 255   # Max value for white pixels
```

```
# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get ndarray reference

# Call the cv_lite extension for binarization
# Returns binary image ndarray
binary_np = cv_lite.grayscale_threshold_binary(image_shape, img_np, thresh,
maxval)

# Wrap ndarray as grayscale image for display
img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
                        alloc=image.ALLOC_REF, data=binary_np)
```

Parameters

Parameter Name	Description	Input/Output	Explanation
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
thresh	Threshold, int type	Input	
maxval	Maximum value, int type	Input	

Return Value

Return Value	Description
binary_np	Binarized image data, ulab.numpy.ndarray type, creates an image instance based on this data

rgb888_threshold_binary

Description

Binarizes an RGB888 image and returns the binarized image data as a ulab.numpy.ndarray. This data can be used to create an Image instance.

Syntax

Please ensure that the output image in the Sensor configuration is RGB888; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640]  # Height, width

# -------------------------------
# Binary threshold parameters
# ------------------------------
thresh = 130   # Threshold value
maxval = 255   # Max value for white pixels

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get ndarray reference

# Call binary threshold function (returns ndarray)
binary_np = cv_lite.rgb888_threshold_binary(image_shape, img_np, thresh, maxval)

# Construct grayscale image for display
img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
                        alloc=image.ALLOC_REF, data=binary_np)
```

Parameters

Parameter Name	Description	Input/Output	Explanation
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
thresh	Threshold, int type	Input	
maxval	Maximum value, int type	Input	

Return Value

Return Value	Description
binary_np	Binarized image data, ulab.numpy.ndarray type, creates an image instance based on this data

rgb888_adjust_exposure

Description

Adjusts the exposure of an RGB888 image and returns the adjusted image data as a ulab.numpy.ndarray. This data can be used to create an Image instance for display.

Syntax

Please ensure that the sensor is configured to output an RGB888 image; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640]  # Height, width

# ------------------------------
# Exposure gain factor (<1 decreases brightness, >1 increases brightness)
# ------------------------------
exposure_gain = 2.5          # Recommended range: 0.2~3.0, 1.0 = no gain

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get RGB888 ndarray reference (HWC)

# Apply exposure adjustment using cv_lite module
exposed_np = cv_lite.rgb888_adjust_exposure(image_shape, img_np, exposure_gain)

# Wrap processed image for display
img_out = image.Image(image_shape[1], image_shape[0], image.RGB888,
                        alloc=image.ALLOC_REF, data=exposed_np)
```

Parameters

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
exposure_gain	Exposure gain factor, float type	Input	

Return Value

Return Value	Description
exposed_np	Image data after exposure adjustment, ulab.numpy.ndarray type, creates an image instance based on this data

rgb888_adjust_exposure_fast

Description

Quickly adjusts the exposure of an RGB888 image, returning the adjusted image data in a ulab.numpy.ndarray. This data can be used to create an Image instance for display. This method is an accelerated version of the software exposure method described above.

Syntax

Please ensure that the sensor is configured to output an RGB888 image; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640]  # Height, width

# ------------------------------
# Exposure adjustment parameters
# exposure_gain: Exposure gain factor, recommended range: 0.2 to 3.0
# A value less than 1.0 reduces exposure (darkening), while a value greater than
1.0 increases exposure (brightening).
# exposure_gain < 1.0: darker, > 1.0: brighter
# -------------------------------
exposure_gain = 2.5  # Example: brighten image by 1.5x

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get RGB888 ndarray (HWC)

# Apply exposure gain using cv_lite
exposed_np = cv_lite.rgb888_adjust_exposure_fast(
    image_shape,
    img_np,
    exposure_gain
)

# Wrap processed image for display
img_out = image.Image(image_shape[1], image_shape[0], image.RGB888,
                        alloc=image.ALLOC_REF, data=exposed_np)
```

Parameters

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
exposure_gain	Exposure gain factor, float type	Input	

Return Value

Return Value	Description
exposed_np	Image data after exposure adjustment, ulab.numpy.ndarray type, creates an image instance based on this data

rgb888_white_balance_gray_world_fast

Description

Quickly white balances an RGB888 image using the grayscale world algorithm. Returns the white-balanced image data in a ulab.numpy.ndarray format, which can be used to create an Image instance for display. This method is an accelerated version of the software white balance method described above.

Syntax

Please ensure that the sensor is configured to output an RGB888 image; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640]  # Height, width

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get RGB888 ndarray reference (HWC)

# Accelerated grayscale world white balancing using cv_lite
# Apply fast gray world white balance
balanced_np = cv_lite.rgb888_white_balance_gray_world_fast(image_shape, img_np)

# Wrap processed image for display
img_out = image.Image(image_shape[1], image_shape[0],
image.RGB888,alloc=image.ALLOC_REF, data=balanced_np)
```

Parameters

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, in the order [height, width], e.g., [480, 640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	

Return Value

Return Value	Description
balanced_np	White-balanced image data, ulab.numpy.ndarray type, used to create an image instance

rgb888_white_balance_gray_world_fast_ex

Description

Quickly white balances an RGB888 image using the grayscale world algorithm, returning the white-balanced image data in a ulab.numpy.ndarray. This data can be used to create an Image instance for display. This method is a parameter-adjustable version of the above method.

Syntax

Please ensure that the sensor is configured to output an RGB888 image; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640]  # Height, width

# ------------------------------
# White balance parameters
# -------------------------------
gain_clip = 2.5             # Gain limit to prevent color blowout
brightness_boost = 1.25     # Global brightness boost

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get RGB888 ndarray reference (HWC)

# Accelerated grayscale world white balance processing using cv_lite (adjustable parameters)
# Apply fast gray world white balance with tunable parameters
balanced_np = cv_lite.rgb888_white_balance_gray_world_fast_ex(
    image_shape,
    img_np,
    gain_clip,
    brightness_boost
)

# Wrap processed image for display
img_out = image.Image(image_shape[1], image_shape[0], image.RGB888,
                        alloc=image.ALLOC_REF, data=balanced_np)
```

Parameter

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
gain_clip	Gain limit coefficient, float type	Input	
brightness_boost	Brightness boost coefficient, float type	Input	

Return Value

Return Value	Description
balanced_np	White-balanced image data, ulab.numpy.ndarray type, from which an image instance can be created for display

rgb888_white_balance_white_patch

Description

Uses the white patch algorithm to white balance an RGB888 image, returning the white-balanced image data as a ulab.numpy.ndarray. This data can be used to create an Image instance for display.

Syntax

Please ensure that the output image of the sensor is configured as an RGB888 image; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640]  # Height, width

# Capture a frame of image
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # RGB888 raw image

# Call the cv_lite extension module for white patch white balancing
balanced_np = cv_lite.rgb888_white_balance_white_patch(image_shape, img_np)

# Construct RGB888 display image
img_out = image.Image(image_shape[1], image_shape[0], image.RGB888,
                        alloc=image.ALLOC_REF, data=balanced_np)
```

Parameters

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, in the order [height, width], e.g., [480, 640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	

Return Value

Return Value	Description
balanced_np	White-balanced image data, ulab.numpy.ndarray type, used to create an image instance

rgb888_white_balance_white_patch_ex

Description

Uses the white patch algorithm to white balance an RGB888 image, returning the white-balanced image data in a ulab.numpy.ndarray. This data can be used to create an Image instance for display. This method is a parameter-adjustable version of the above method.

Syntax

Please ensure that the sensor is configured to output an RGB888 image; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640]  # Height, width
```

```
# -------------------------------
# White balance parameters
# -------------------------------
top_percent = 5.0          # Percentage of brightest pixels used for white point estimation
gain_clip = 2.5            # Maximum gain limit
brightness_boost = 1.1     # Brightening Factor

# Capture one frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get image data reference (ulab ndarray)

# Perform white point white balancing
balanced_np = cv_lite.rgb888_white_balance_white_patch_ex(
    image_shape, img_np,
    top_percent,
    gain_clip,
    brightness_boost
)

# Wrapping images for display
img_out = image.Image(image_shape[1], image_shape[0], image.RGB888,
                        alloc=image.ALLOC_REF, data=balanced_np)
```

Parameter

Parameter Name	Description	Input/Output	Note
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
top_percent	Percentage of the brightest pixels used for white point estimation, float type	Input	
gain_clip	Maximum gain limit, float type	Input	
brightness_boost	Brightness boost factor, float type	Input	

Return Value

Return Value	Description
balanced_np	Image data after white balance, ulab.numpy.ndarray type. Create an image instance based on this data for display

rgb888_erode

Description

Applies an erosion operation to an RGB888 image, returning the eroded single-channel image data as a ulab.numpy.ndarray. This data can be used to create an Image instance for display.

Syntax

Please ensure that the output image in the Sensor configuration is RGB888; otherwise, an error will occur.

```
import cv_lite

# -------------------------------
# Corrosion algorithm parameter settings
# -------------------------------
kernel_size = 3         # Convolution kernel size (must be an odd number, such as 3, 5, 7)
iterations = 1          # Erosion iterations
threshold_value = 100   # Binarization threshold (0 = use Otsu automatic thresholding)

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get a reference to an RGB888 ndarray

# Apply an erosion operation (after automatic conversion to grayscale)
eroded_np = cv_lite.rgb888_erode(
    image_shape,
    img_np,
    kernel_size,
    iterations,
    threshold_value
)

# Construct an image for display (grayscale format)
img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
                        alloc=image.ALLOC_REF, data=eroded_np)
```

Parameter

Parameter Name	Description	Input/Output	Note
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
kernel_size	Convolution kernel size, int type	Input	
iterations	Number of erosion iterations, int type	Input	
threshold_value	Binarization threshold, int type	Input	

Return Value

Return Value	Description
eroded_np	Image data after erosion, ulab.numpy.ndarray type

rgb888_dilate

Description

Applies a dilation operation to an RGB888 image, returning the dilated single-channel image data as a ulab.numpy.ndarray. This data can be used to create an Image instance for display.

Syntax

Please ensure that the output image of the sensor is configured as RGB888; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640] # Height, width

# -------------------------------
# Dilation algorithm parameter settings
# -------------------------------
kernel_size = 3         # Convolution kernel size (Odd numbers are recommended, but some implementations support even numbers)
iterations = 1          # Dilation iterations
threshold_value = 100   # Binarization threshold (0 = use Otsu automatic thresholding)

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # 获取 RGB888 ndarray 引用 / Get RGB888 ndarray reference

# Apply dilation (convert to grayscale internally before dilation)
result_np = cv_lite.rgb888_dilate(
    image_shape,
    img_np,
    kernel_size,
    iterations,
    threshold_value
)

# Construct an image for display (grayscale format)
img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
                        alloc=image.ALLOC_REF, data=result_np)
```

Parameter

Parameter Name	Description	Input/Output	Note
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
kernel_size	Convolution kernel size, int type	Input	
iterations	Number of dilation iterations, int type	Input	
threshold_value	Binarization threshold, int type	Input	

Return Value

Return Value	Description
dilated_np	Dilated image data, ulab.numpy.ndarray type

rgb888_open

Description

Performs an opening operation on an RGB888 image, returning the resulting single-channel image data as a ulab.numpy.ndarray. This data can be used to create an image instance for display.

Syntax

Please ensure that the output image in the sensor configuration is RGB888; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640] # Height, width

# ------------------------------
# Open operation parameter settings
# ------------------------------
kernel_size = 3         # Convolution kernel size (should be an odd number)
iterations = 1          # Number of operation iterations
threshold_value = 100   # Binarization threshold (0 = use Otsu automatic
thresholding)

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get a reference to an RGB888 ndarray

# Perform an opening operation (erode first and then dilate)
open_np = cv_lite.rgb888_open(
    image_shape,
    img_np,
    kernel_size,
    iterations,
    threshold_value
)

# Construct a grayscale image object for display
img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
                        alloc=image.ALLOC_REF, data=open_np)
```

Parameter

Parameter Name	Description	Input/Output	Note
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
kernel_size	Convolution kernel size, int type	Input	
iterations	Number of opening iterations, int type	Input	
threshold_value	Binarization threshold, int type	Input	

Return Value

Return Value	Description
opened_np	Image data after opening, ulab.numpy.ndarray type

rgb888_close

Description

Applies a closed RGB888 image, returning the closed single-channel ulab.numpy.ndarray image data. This data can be used to create an Image instance for display.

Syntax

Please ensure that the output image of the sensor is configured as RGB888; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640] # Height, width

# ------------------------------
# Closed operation parameter settings
# ------------------------------
kernel_size = 3         # Convolution kernel size (odd number recommended)
iterations = 1          # Number of operation iterations
threshold_value = 100   # Binarization threshold (0 = use Otsu automatic thresholding)

# Get image frame
img = sensor.snapshot()
```

```
img_np = img.to_numpy_ref()  # Get a reference to an RGB888 ndarray

# Apply a closing operation (dilation followed by erosion)
closed_np = cv_lite.rgb888_close(
    image_shape,
    img_np,
    kernel_size,
    iterations,
    threshold_value
)

# Construct an image object (grayscale image) for display
img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
                        alloc=image.ALLOC_REF, data=closed_np)
```

Parameter

Parameter Name	Description	Input/Output	Note
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
kernel_size	Convolution kernel size, int type	Input	
iterations	Number of closing iterations, int type	Input	
threshold_value	Binarization threshold, int type	Input	

Return Value

Return Value	Description
closed_np	Image data after closing operation, ulab.numpy.ndarray type

rgb888_tophat

Description

Performs a top-hat operation on an RGB888 image, returning the resulting single-channel ulab.numpy.ndarray image data. You can create an image instance based on this data for display.

Syntax

Please ensure that the output image in the Sensor configuration is RGB888; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640] # Height, width

# -------------------------------
# Top hat operation parameter settings
# -------------------------------
kernel_size = 3         # Convolution kernel size (odd number recommended)
iterations = 1          # Number of operation iterations
threshold_value = 100   # Binarization threshold (0 = use Otsu automatic
thresholding)

# Get a frame of image and convert it into ndarray
img = sensor.snapshot()
img_np = img.to_numpy_ref()

# Perform top-hat operation
# Top hat = original image - opening operation result
tophat_np = cv_lite.rgb888_tophat(
    image_shape,
    img_np,
    kernel_size,
    iterations,
    threshold_value
)

# Construct an image object and display it
img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
                        alloc=image.ALLOC_REF, data=tophat_np)
```

Parameter

Parameter Name	Description	Input/Output	Note
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
kernel_size	Convolution kernel size, int type	Input	
iterations	Number of top-hat operation iterations, int type	Input	
threshold_value	Binarization threshold, int type	Input	

Return Value

Return Value	Description
tophat_np	Image data after top-hat operation, ulab.numpy.ndarray type

rgb888_blackhat

Description

Performs a blackhat operation on an RGB888 image, returning the resulting single-channel image data as a ulab.numpy.ndarray. This data can be used to create an Image instance for display.

Syntax

Please ensure that the output image in the Sensor configuration is RGB888; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640] # Height, width

# -------------------------------
# Black hat operation parameter settings
# -------------------------------
kernel_size = 3         # Convolution kernel size (odd number recommended)
iterations = 1          # Number of operation iterations
threshold_value = 100   # Binarization threshold (0 = use Otsu automatic
thresholding)

# Capture a frame and convert to ndarray
img = sensor.snapshot()
img_np = img.to_numpy_ref()

# Perform black hat operations
# Black hat = closing operation - original image
blackhat_np = cv_lite.rgb888_blackhat(
    image_shape,
    img_np,
    kernel_size,
    iterations,
    threshold_value
)

# Construct an image object and display it
img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
                        alloc=image.ALLOC_REF, data=blackhat_np)
```

Parameter

Parameter Name	Description	Input/Output	Note
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
kernel_size	Convolution kernel size, int type	Input	
iterations	Number of blackhat operation iterations, int type	Input	
threshold_value	Binarization threshold, int type	Input	

Return Value

Return Value	Description
blackhat_np	Image data after blackhat operation, ulab.numpy.ndarray type

rgb888_gradient

Description

Applies a morphological gradient operation to an RGB888 image, returning a single-channel ulab.numpy.ndarray image data after the morphological gradient. This data can be used to create an Image instance for display.

Syntax

Please ensure that the output image in the Sensor configuration is RGB888; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640] # Height, width

# ================================
# Gradient operating parameters
# ================================
kernel_size = 3        # Convolution kernel size (odd number recommended)
iterations = 1         # Morphological iterations
threshold_value = 100  # Binarization threshold (0 = use Otsu automatic thresholding)

# Capture image and convert to ndarray
img = sensor.snapshot()
img_np = img.to_numpy_ref()

# Call the gradient operation (Gradient = dilation - erosion)
gradient_np = cv_lite.rgb888_gradient(image_shape, img_np, kernel_size, iterations, threshold_value)

# Construct a grayscale image for display
img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
                        alloc=image.ALLOC_REF, data=gradient_np)
```

Parameter

Parameter Name	Description	Input/Output	Note
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
kernel_size	Convolution kernel size, int type	Input	
iterations	Number of morphological iterations, int type	Input	
threshold_value	Binarization threshold, int type	Input	

Return Value

Return Value	Description
gradient_np	Image data after morphological gradient, ulab.numpy.ndarray type

rgb888_mean_blur

Description

Applies a mean blur to an RGB888 image, returning the blurred image data as a ulab.numpy.ndarray. This data can be used to create an Image instance for display.

Syntax

Please ensure that the output image in the Sensor configuration is RGB888; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640] # Height, width

# -------------------------------
# Mean blur kernel size
# Must be odd: 3, 5, 7, etc.
# -------------------------------
kernel_size = 3

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get an RGB888 image reference

# Apply mean fuzzy filtering
mean_blur_np = cv_lite.rgb888_mean_blur_fast(
    image_shape,
    img_np,
    kernel_size
)

# Constructing an image for display
```

```
img_out = image.Image(image_shape[1], image_shape[0], image.RGB888,
                      alloc=image.ALLOC_REF, data=mean_blur_np)
```

Parameter

Parameter Name	Description	Input/Output	Note
image_shape	Image shape, list type, in the order [height, width], e.g., [480, 640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
kernel_size	Convolution kernel size, int type	Input	

Return Value

Return Value	Description
mean_blur_np	Image data after mean blurring, ulab.numpy.ndarray type

rgb888_gaussian_blur

Description

Applies a Gaussian blur to an RGB888 image, returning the blurred image data as a ulab.numpy.ndarray. This data can be used to create an Image instance for display.

Syntax

Please ensure that the output image in the Sensor configuration is RGB888; otherwise, an error will occur.

```
import cv_lite
import image

image_shape = [480,640] # Height, width

# -------------------------------
# Gaussian filter kernel size
# Must be odd: 3, 5, 7, etc.
# -------------------------------
kernel_size = 3

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get an RGB888 image reference

# Apply a Gaussian blur filter
gaussian_blur_np = cv_lite.rgb888_gaussian_blur_fast(
    image_shape,
    img_np,
    kernel_size
)
```

```
# Constructing an image for display
img_out = image.Image(image_shape[1], image_shape[0], image.RGB888,
                        alloc=image.ALLOC_REF, data=gaussian_blur_np)
```

Parameters

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, in the order [height, width], e.g., [480,640]	Input	
img_np	Image data reference, ulab.numpy.ndarray type	Input	
kernel_size	Convolution kernel size, int type	Input	

Return Value

Return Value	Description
gaussian_blur_np	Image data after Gaussian blur, ulab.numpy.ndarray type

rgb888_calc_histogram

Description

Calculates the histogram of an RGB888 image and returns the histogram data.

Syntax

```
import cv_lite

# Capture a frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get an RGB888 image reference

# Use cv_lite to calculate the RGB histogram (return an array of shape 3x256)
hist = cv_lite.rgb888_calc_histogram(image_shape, img_np)
```

Parameter

Parameter Name	Description	Input/Output	Notes
image_shape	Image shape, list type, including width and height, such as [1920, 1080]	Input	
img_np	Image data reference	Input	

Return Value

Return Value	Description
hist	Histogram data

rgb888_find_corners

Description

Finds corner points in an RGB888 image and returns their coordinates.

Syntax

Please ensure that the sensor is configured to output an RGB888 image; otherwise, an error will occur.

```python
import cv_lite

image_shape = [240,320] # Height, width

# -------------------------------
# Adjustable parameters (recommended for debugging)
# -------------------------------
max_corners       = 20       # Maximum number of corner points
quality_level     = 0.01     # Shi-Tomasi quality factor
min_distance      = 20.0     # Minimum corner distance

# Capture current frame
img = sensor.snapshot()
img_np = img.to_numpy_ref()  # Get a reference to an RGB888 ndarray

# Call the corner detection function and return the corner point array [x0, y0, x1, y1, ...]
corners = cv_lite.rgb888_find_corners(
    image_shape, img_np,
    max_corners,
    quality_level,
    min_distance
)

# Traverse the corner point array and draw the corner points
for i in range(0, len(corners), 2):
    x = corners[i]
    y = corners[i + 1]
    img.draw_circle(x, y, 3, color=(0, 255, 0), fill=True)
```

Parameter

Parameter Name	Description	Input/Output	Note
image_shape	Image shape, list type, including width and height, such as [240,320]	Input	
img_np	Image data reference	Input	
max_corners	Maximum number of corner points	Input	
quality_level	Shi-Tomasi quality factor	Input	
min_distance	Minimum distance	Input	

Return Value

Return Value	Description
corners	Corner point coordinate array, where each two numbers represent the coordinates of a corner point, such as [x0, y0, x1, y1, ...]