

ImageProcess API

Contents

1	File Documentation	1
1.1	image_proc.h File Reference	1
1.1.1	Detailed Description	6
1.1.2	Macro Definition Documentation	6
1.1.2.1	ROTATE_180_CLOCKWISE	6
1.1.2.2	ROTATE_270_CLOCKWISE	6
1.1.2.3	ROTATE_90_CLOCKWISE	6
1.1.3	Typedef Documentation	6
1.1.3.1	RGBAffineInfo_t	6
1.1.4	Function Documentation	7
1.1.4.1	BatchRGBAffine(uint8_t *input, uint8_t *output, int batch, int out_batch_offset, int in_w, int in_h, RGBAffineInfo_t *param, cudaStream_t stream)	7
1.1.4.2	BatchRGBBilinearResizeNormPadPlane(uint8_t *in_buf, float *out_buf, int batch, int input_batch_offset, int output_batch_offset, batch_resize_param *param, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, float pad1, float pad2, float pad3, bool channel_rev, cudaStream_t stream)	7
1.1.4.3	FullRangeNV12ToRGBBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	8
1.1.4.4	FullRangeYU12ToRGBBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	8
1.1.4.5	GrayAffine(uint8_t *input, uint8_t *output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], cudaStream_t stream)	9
1.1.4.6	GrayResizeBilinear(uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)	9

1.1.4.7	GrayResizeNearest(uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)	10
1.1.4.8	MultiRoiRGBBilinearResizeNormPadPlane(uint8_t *in_buf, float *out_buf, int batch, int in_h, int in_w, int output_batch_offset, multi_roi_resize_param *param, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, float pad1, float pad2, float pad3, bool channel_rev, cudaStream_t stream)	10
1.1.4.9	non_max_suppression(float *predict, int num_batch, int num_bboxes, int num←_classes, float confidence_threshold, float nms_threshold, float *pout, int max←_objects, cudaStream_t stream)	11
1.1.4.10	NV12Crop(uint8_t *in_buf, uint8_t *out_buf, int start_w, int start_h, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)	11
1.1.4.11	NV12ToBGRBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	11
1.1.4.12	NV12ToBGRNearestResizeNormPlane(uint8_t *in_buf, float *out_buf, int in←_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	12
1.1.4.13	NV12ToRGBBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	13
1.1.4.14	NV12ToRGBNearestResizeNormPlane(uint8_t *in_buf, float *out_buf, int in←_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	13
1.1.4.15	RadixSortFunc(DataType *data_dev, IndxType *indx←_dev, int32_t total←_num, DataType *data_out_dev=nullptr, IndxType *indx←_out_dev=nullptr, void *workspace_dev=nullptr, cudaStream_t *stream=nullptr)	14
1.1.4.16	RGB2YU12(uint8_t *in_buf, uint8_t *out_buf, int w, int h, cudaStream_t stream)	14
1.1.4.17	RGBAffine(uint8_t *input, uint8_t *output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], cudaStream←_t stream)	14
1.1.4.18	RGBBilinearResizeCropNormPlaneV2(uint8_t *in_buf, float *out_buf, uchar4 *ws, int in_w, int in_h, int resized_w, int resized_h, int crop_w_start, int crop_h←_start, int crop_w, int crop_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, bool fmt_cvt, cudaStream_t stream)	15
1.1.4.19	RGBCrop(uint8_t *in_buf, uint8_t *out_buf, int start_w, int start_h, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)	15
1.1.4.20	RGBNormalization(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int in_c, float mean, float standard, float scale, cudaStream_t stream)	16
1.1.4.21	RGBNormalization_3Channels(uint8_t *in_buf, float *out_buf, int in_w, int in_h, float mean1, float mean2, float mean3, float standard1, float standard2, float stan←_dard3, float scale, bool input_plane, bool output_plane, bool channel_rev, cuda←_Stream_t stream)	16

1.1.4.22	<code>RGBResizeBilinear(uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)</code>	17
1.1.4.23	<code>RGBResizeNearest(uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)</code>	17
1.1.4.24	<code>RGBResizePlaneBilinear(uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)</code>	17
1.1.4.25	<code>RGBResizePlaneNearest(uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)</code>	18
1.1.4.26	<code>RGBResizePlanePadBilinear(uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, int w_box, int h_box, int w_b, int h_b, cudaStream_t stream)</code>	18
1.1.4.27	<code>RGBResizePlanePadNearest(uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, int w_box, int h_box, int w_b, int h_b, cudaStream_t stream)</code>	19
1.1.4.28	<code>RGBResizeWithROIBilinear(uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, int roi_w_start, int roi_h_start, int roi_w, int roi_h, cudaStream_t stream)</code>	19
1.1.4.29	<code>RGBResizeWithROINearest(uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, int roi_w_start, int roi_h_start, int roi_w, int roi_h, cudaStream_t stream)</code>	19
1.1.4.30	<code>RGBROIBilinearResizeNormPadPlane(uint8_t *in_buf, float *out_buf, int w_in, int h_in, int w_out, int h_out, int img_w, int img_h, int pad_w, int pad_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, float pad1, float pad2, float pad3, bool channel_rev, cudaStream_t stream)</code>	20
1.1.4.31	<code>RGBROIBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int w_in, int h_in, int w_out, int h_out, int roi_w_start, int roi_h_start, int roi_w, int roi_h, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, bool channel_rev, cudaStream_t stream)</code>	21
1.1.4.32	<code>RGBROINearestResizeNormPadPlane(uint8_t *in_buf, float *out_buf, int w_in, int h_in, int w_out, int h_out, int img_w, int img_h, int pad_w, int pad_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, float pad1, float pad2, float pad3, bool channel_rev, cudaStream_t stream)</code>	21
1.1.4.33	<code>RGBROINearestResizeNormPlane(uint8_t *in_buf, float *out_buf, int w_in, int h_in, int w_out, int h_out, int roi_w_start, int roi_h_start, int roi_w, int roi_h, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, bool channel_rev, cudaStream_t stream)</code>	22
1.1.4.34	<code>RGBRotate(uint8_t *in, uint8_t *out, int in_w, int in_h, int rotate_mode, cudaStream_t stream)</code>	23
1.1.4.35	<code>RoiNv122RGBAffineNorm(uint8_t *input, float *output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)</code>	23
1.1.4.36	<code>RoiNV12ToBGRBilinearResizePlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, cudaStream_t stream)</code>	23

1.1.4.37	RoiNV12ToRGBBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	24
1.1.4.38	RoiNV12ToRGBBilinearResizePlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, cudaStream_t stream)	25
1.1.4.39	RoiNV12ToRGBBilinearResizeQuantizePlane(uint8_t *in_buf, uint8_t *out_buf, uchar4 *ws, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, float zero_point, float scales_input, cudaStream_t stream)	25
1.1.4.40	RoiYU122BGRAffineNorm(uint8_t *input, float *output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	26
1.1.4.41	RoiYU122RGBAffineNorm(uint8_t *input, float *output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	27
1.1.4.42	RoiYU12ToBGRBilinearResizePlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, cudaStream_t stream)	27
1.1.4.43	RoiYU12ToRGBBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	28
1.1.4.44	RoiYU12ToRGBBilinearResizeNormPlaneV2(uint8_t *in_buf, float *out_buf, uchar4 *ws, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, int input_pad_w, int input_pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, uint8_t pad1, uint8_t pad2, uint8_t pad3, cudaStream_t stream)	29
1.1.4.45	RoiYU12ToRGBBilinearResizePlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, cudaStream_t stream)	29
1.1.4.46	RoiYU12ToRGBBilinearResizeQuantizePlane(uint8_t *in_buf, uint8_t *out_buf, uchar4 *ws, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, float zero_point, float scales_input, cudaStream_t stream)	30
1.1.4.47	RoiYUV400PToRGBBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean, float std, float scale, float pad, cudaStream_t stream)	31

1.1.4.48	RoiYUV422PToRGBBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	31
1.1.4.49	RoiYUV422ToRGBBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	32
1.1.4.50	RoiYUV444PToRGBBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	33
1.1.4.51	YU122RGBAffine(uint8_t *input, uint8_t *output, int in_w, int in_h, int out_w, int out_h, float m[6], cudaStream_t stream)	33
1.1.4.52	YU12Crop(uint8_t *in_buf, uint8_t *out_buf, int start_w, int start_h, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)	34
1.1.4.53	YU12ToBGRBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	34
1.1.4.54	YU12ToBGRNearestResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	35
1.1.4.55	YU12ToRGBBilinearResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	35
1.1.4.56	YU12ToRGBNearestResizeNormPlane(uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)	36
1.1.4.57	YUV440pToYUV420p(uint8_t *sy, uint8_t *su, uint8_t *sv, uint8_t *dy, uint8_t *du, uint8_t *dv, int w, int h, int align_w, cudaStream_t stream)	36
1.1.4.58	YUV444pToYUV420p(uint8_t *sy, uint8_t *su, uint8_t *sv, uint8_t *dy, uint8_t *du, uint8_t *dv, int w, int h, int align_w, cudaStream_t stream)	37
1.1.4.59	YUVI420ResizeBilinear(uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)	37
1.1.4.60	YUVI420ResizeNearest(uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)	37
1.1.4.61	YUVNv12ResizeBilinear(uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)	38

1.1.4.62	<code>YUVNv12ResizeNearest(uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)</code>	38
1.1.4.63	<code>YUVNv12ToRGB(uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, cudaStream_t stream)</code>	38
1.1.4.64	<code>YUVNv12ToRGBFloat(uint8_t *in_buf, float *out_buf, int in_w, int in_h, cudaStream_t stream)</code>	39
1.1.4.65	<code>YUVNv12ToRGBPlane(uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, cudaStream_t stream)</code>	39
1.1.4.66	<code>YUVNv21ResizeBilinear(uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)</code>	39
1.1.4.67	<code>YUVNv21ResizeNearest(uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)</code>	40
1.1.4.68	<code>YUVYu12ToRGB(uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, cudaStream_t stream)</code>	40
1.1.4.69	<code>YUVYu12ToRGBFloat(uint8_t *in_buf, float *out_buf, int in_w, int in_h, cudaStream_t stream)</code>	40
1.1.4.70	<code>YUVYu12ToRGBPlane(uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, cudaStream_t stream)</code>	40
Index		43

Chapter 1

File Documentation

1.1 image_proc.h File Reference

Full APIs which support in this demo.

Classes

- struct [RGBAffineInfo](#)
Affine transformation info structure.
- struct [batch_resize_param](#)
multiple batch resize info structure.
- struct [multi_roi_resize_param](#)
multiple roi resize info structure.

Macros

- #define [ROTATE_90_CLOCKWISE](#) 0
- #define [ROTATE_180_CLOCKWISE](#) 1
- #define [ROTATE_270_CLOCKWISE](#) 2

Typedefs

- typedef struct [RGBAffineInfo](#) [RGBAffineInfo_t](#)
Affine transformation info structure.

Functions

- void [RGBResizeBilinear](#) (uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)
Resize RGB image with bilinear interpolation.
- void [RGBResizeNearest](#) (uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)
Resize RGB image with nearest neighbor interpolation.
- void [GrayResizeBilinear](#) (uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)
Resize gray image with bilinear interpolation.
- void [GrayResizeNearest](#) (uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)
Resize gray image with nearest neighbor interpolation.
- void [YUVNv12ResizeNearest](#) (uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)
Resize YUV(nv12) image with nearest neighbor interpolation.
- void [YUVNv21ResizeNearest](#) (uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)
Resize YUV(nv21) image with nearest neighbor interpolation.
- void [YUVI420ResizeNearest](#) (uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)
Resize YUV(i420) image with nearest neighbor interpolation.
- void [YUVNv12ResizeBilinear](#) (uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)
Resize YUV(nv12) image with bilinear interpolation.
- void [YUVNv21ResizeBilinear](#) (uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)
Resize YUV(nv21) image with bilinear interpolation.
- void [YUVI420ResizeBilinear](#) (uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, int out_w, int out_h, cudaStream_t stream)
Resize YUV(i420) image with bilinear interpolation.
- void [YUVYu12ToRGB](#) (uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, cudaStream_t stream)
Convert YUV(yu12, also called i420) image to RGB image.
- void [YUVNv12ToRGB](#) (uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, cudaStream_t stream)
Convert YUV(nv12) image to RGB image.
- void [YUVYu12ToRGBFloat](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, cudaStream_t stream)
Convert YUV(yu12, also called i420) image to RGB image.
- void [YUVNv12ToRGBFloat](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, cudaStream_t stream)
Convert YUV(nv12) image to RGB image.
- void [YUVYu12ToRGBPlane](#) (uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, cudaStream_t stream)
Convert YUV(yu12, also called i420) image to RGB image.
- void [YUVNv12ToRGBPlane](#) (uint8_t *in_buf, uint8_t *out_buf, int in_w, int in_h, cudaStream_t stream)
Convert YUV(nv12) image to RGB image.
- void [RGBResizeWithROINearest](#) (uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, int roi_w_start, int roi_h_start, int roi_w, int roi_h, cudaStream_t stream)
Resize RGB image ROI area with nearest interpolation.
- void [RGBResizeWithROIBilinear](#) (uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, int roi_w_start, int roi_h_start, int roi_w, int roi_h, cudaStream_t stream)
Resize RGB image ROI area with bilinear interpolation.
- void [RGBResizePlaneBilinear](#) (uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)
Resize RGB image with bilinear interpolation.

- void [RGBResizePlaneNearest](#) (uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)
Resize RGB image with nearest interpolation.
- void [RGBResizePlanePadBilinear](#) (uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, int w_box, int h_box, int w_b, int h_b, cudaStream_t stream)
Resize RGB image with bilinear interpolation.
- void [RGBResizePlanePadNearest](#) (uint8_t *in_buf, uint8_t *out_buf, int w_in, int h_in, int w_out, int h_out, int w_box, int h_box, int w_b, int h_b, cudaStream_t stream)
Resize RGB image with nearest interpolation.
- void [RGBCrop](#) (uint8_t *in_buf, uint8_t *out_buf, int start_w, int start_h, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)
Crop RGB image.
- void [YU12Crop](#) (uint8_t *in_buf, uint8_t *out_buf, int start_w, int start_h, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)
Crop YU12 image.
- void [NV12Crop](#) (uint8_t *in_buf, uint8_t *out_buf, int start_w, int start_h, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)
Crop NV12 image.
- void [RGBNormalization](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int in_c, float mean, float standard, float scale, cudaStream_t stream)
RGB image normalization.
- void [RGBNormalization_3Channels](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, float mean1, float mean2, float mean3, float standard1, float standard2, float standard3, float scale, bool input_plane, bool output_plane, bool channel_rev, cudaStream_t stream)
RGB image normalization.
- void [NV12ToRGBBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
NV12 image convert to RGB and resize with padding and normalization.
- void [YU12ToRGBBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
YU12(i420) image convert to RGB and resize with padding and normalization.
- void [NV12ToRGBNearestResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
NV12 image convert to RGB and resize with padding and normalization.
- void [YU12ToRGBNearestResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
YU12(i420) image convert to RGB and resize with padding and normalization.
- void [NV12ToBGRBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
NV12 image convert to BGR and resize with padding and normalization.
- void [YU12ToBGRBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
YU12(i420) image convert to BGR and resize with padding and normalization.
- void [NV12ToBGRNearestResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
NV12 image convert to BGR and resize with padding and normalization.

- void [YU12ToBGRNearestResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
YU12(i420) image convert to BGR and resize with padding and normalization.
- void [RGBROI NearestResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int w_in, int h_in, int w_out, int h_out, int roi_w_start, int roi_h_start, int roi_w, int roi_h, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, bool channel_rev, cudaStream_t stream)
Resize RGB/BGR image ROI area with nearest interpolation and normalization.
- void [RGBROI BilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int w_in, int h_in, int w_out, int h_out, int roi_w_start, int roi_h_start, int roi_w, int roi_h, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, bool channel_rev, cudaStream_t stream)
Resize RGB/BGR image ROI area with bilinear interpolation and normalization.
- void [RoiNV12ToRGBBilinearResizePlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, cudaStream_t stream)
NV12 image roi area convert to RGB and resize with padding.
- void [RoiYU12ToRGBBilinearResizePlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, cudaStream_t stream)
YU12(i420) image roi area convert to RGB and resize with padding and normalization.
- void [RoiNV12ToBGRBilinearResizePlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, cudaStream_t stream)
NV12 image roi area convert to BGR and resize with padding.
- void [RoiYU12ToBGRBilinearResizePlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, cudaStream_t stream)
YU12(i420) image roi area convert to BGR and resize with padding and normalization.
- void [RoiNV12ToRGBBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
NV12 image roi area convert to RGB and resize with padding and normalization.
- void [RoiYU12ToRGBBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
YU12(i420) image roi area convert to RGB and resize with padding and normalization.
- void [FullRangeNV12ToRGBBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
NV12 full range(0~255) image convert to RGB and resize with padding and normalization.
- void [FullRangeYU12ToRGBBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)
YU12(i420) full range(0~255) image convert to RGB and resize with padding and normalization.
- void [GrayAffine](#) (uint8_t *input, uint8_t *output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], cudaStream_t stream)
Affine transformation gray image with bilinear interpolation.
- void [RGBAffine](#) (uint8_t *input, uint8_t *output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], cudaStream_t stream)
Affine transformation rgb image with bilinear interpolation.
- void [YU122RGBAffine](#) (uint8_t *input, uint8_t *output, int in_w, int in_h, int out_w, int out_h, float m[6], cudaStream_t stream)

YU12(i420) image convert to RGB and affine transformation.

- void [RoiYUV444PToRGBBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)

YUV444P image roi area convert to RGB and bilinear resize with padding and normalization.

- void [RoiYUV400PToRGBBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean, float std, float scale, float pad, cudaStream_t stream)

YUV400P image roi area convert to RGB and bilinear resize with padding and normalization.

- void [RoiYUV422PToRGBBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)

YUV422P image roi area convert to RGB and bilinear resize with padding and normalization.

- void [RoiYUV422ToRGBBilinearResizeNormPlane](#) (uint8_t *in_buf, float *out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)

YUV422 image roi area convert to RGB and bilinear resize with padding and normalization.

- void [RoiNv122RGBAffineNorm](#) (uint8_t *input, float *output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)

NV12 image roi area convert to RGB and affine transformation with padding and normalization.

- void [RoiYU122RGBAffineNorm](#) (uint8_t *input, float *output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)

YU12 image roi area convert to RGB and affine transformation with padding and normalization.

- void [RoiYU122BGRAffineNorm](#) (uint8_t *input, float *output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)

YU12 image roi area convert to BGR and affine transformation with padding and normalization.

- void [RoiNV12ToRGBBilinearResizeQuantizePlane](#) (uint8_t *in_buf, uint8_t *out_buf, uchar4 *ws, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, float zero_point, float scales_input, cudaStream_t stream)

NV12 image roi area convert to RGB, bilinear resize, padding and quantize output to uint8.

- void [RoiYU12ToRGBBilinearResizeQuantizePlane](#) (uint8_t *in_buf, uint8_t *out_buf, uchar4 *ws, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, float zero_point, float scales_input, cudaStream_t stream)

YU12 image roi area convert to RGB, bilinear resize, padding and quantize output to uint8.

- void [RoiYU12ToRGBBilinearResizeNormPlaneV2](#) (uint8_t *in_buf, float *out_buf, uchar4 *ws, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, int input_pad_w, int input_pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, uint8_t pad1, uint8_t pad2, uint8_t pad3, cudaStream_t stream)

YU12(i420) image roi area convert to RGB and resize(compatible with transforms.Resize of torchvision) with padding and normalization.

- void [RGB2YU12](#) (uint8_t *in_buf, uint8_t *out_buf, int w, int h, cudaStream_t stream)

RGB image convert to YUV420p(i420).

- template<typename DataType, typename IndxType, bool IsAscending = false>
int32_t [RadixSortFunc](#) (DataType *data_dev, IndxType *indx_dev, int32_t total_num, DataType *data_out_dev, IndxType *indx_out_dev, void *workspace_dev, cudaStream_t *stream=nullptr)

Sort data with radix sort algorithm.

- void [non_max_suppression](#) (float *predict, int num_batch, int num_bboxes, int num_classes, float confidence_threshold, float nms_threshold, float *pout, int max_objects, cudaStream_t stream)

Non max suppression inner class.

- void [YUV444pToYUV420p](#) (uint8_t *sy, uint8_t *su, uint8_t *sv, uint8_t *dy, uint8_t *du, uint8_t *dv, int w, int h, int align_w, cudaStream_t stream)
yuv444p transform to yuv420p.
- void [YUV440pToYUV420p](#) (uint8_t *sy, uint8_t *su, uint8_t *sv, uint8_t *dy, uint8_t *du, uint8_t *dv, int w, int h, int align_w, cudaStream_t stream)
yuv440p transform to yuv420p.
- void [RGBROINearestResizeNormPadPlane](#) (uint8_t *in_buf, float *out_buf, int w_in, int h_in, int w_out, int h_out, int img_w, int img_h, int pad_w, int pad_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, float pad1, float pad2, float pad3, bool channel_rev, cudaStream_t stream)
Resize RGB/BGR image ROI area with nearest interpolation and normalization.
- void [RGBROIBilinearResizeNormPadPlane](#) (uint8_t *in_buf, float *out_buf, int w_in, int h_in, int w_out, int h_out, int img_w, int img_h, int pad_w, int pad_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, float pad1, float pad2, float pad3, bool channel_rev, cudaStream_t stream)
Resize RGB/BGR image ROI area with bilinear interpolation and normalization.
- void [RGBBilinearResizeCropNormPlaneV2](#) (uint8_t *in_buf, float *out_buf, uchar4 *ws, int in_w, int in_h, int resized_w, int resized_h, int crop_w_start, int crop_h_start, int crop_w, int crop_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, bool fmt_cvt, cudaStream_t stream)
RGB resize(compatible with transforms.Resize of torchvision) with crop and normalization.
- void [RGBRotate](#) (uint8_t *in, uint8_t *out, int in_w, int in_h, int rotate_mode, cudaStream_t stream)
Rotate RGB image with specific rotate mode.
- void [BatchRGBAffine](#) (uint8_t *input, uint8_t *output, int batch, int out_batch_offset, int in_w, int in_h, [RGBAffineInfo_t](#) *param, cudaStream_t stream)
Affine transformation batch of rgb image with bilinear interpolation.
- void [BatchRGBBilinearResizeNormPadPlane](#) (uint8_t *in_buf, float *out_buf, int batch, int input_batch_offset, int output_batch_offset, [batch_resize_param](#) *param, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, float pad1, float pad2, float pad3, bool channel_rev, cudaStream_t stream)
Multiple batch resize RGB/BGR image with bilinear interpolation and normalization.
- void [MultiRoiRGBBilinearResizeNormPadPlane](#) (uint8_t *in_buf, float *out_buf, int batch, int in_h, int in_w, int output_batch_offset, [multi_roi_resize_param](#) *param, float scale, float mean1, float mean2, float mean3, float std1, float std2, float std3, float pad1, float pad2, float pad3, bool channel_rev, cudaStream_t stream)
Multiple roi resize RGB/BGR image ROI area with bilinear interpolation and normalization.

1.1.1 Detailed Description

Full APIs which support in this demo.

1.1.2 Macro Definition Documentation

1.1.2.1 `#define ROTATE_180_CLOCKWISE 1`

1.1.2.2 `#define ROTATE_270_CLOCKWISE 2`

1.1.2.3 `#define ROTATE_90_CLOCKWISE 0`

1.1.3 Typedef Documentation

1.1.3.1 `typedef struct RGBAffineInfo RGBAffineInfo_t`

Affine transformation info structure.

Parameters

<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>m[6]</i>	Affine transformation matrix.

1.1.4 Function Documentation

1.1.4.1 void BatchRGBAffine (uint8_t * *input*, uint8_t * *output*, int *batch*, int *out_batch_offset*, int *in_w*, int *in_h*, RGBAffineInfo_t * *param*, cudaStream_t *stream*)

Affine transformation batch of rgb image with bilinear interpolation.

This API is suitable for a lot of small rgb images compare with RGBAffine API.

Parameters

<i>input</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3) * sizeof(uint8_t).
<i>output</i>	The output buffer allocate in device memory. The size is (out_w * out_h * 3 * batch) * sizeof(uint8_t).
<i>batch</i>	Number of input images.
<i>out_batch_offset</i>	The offset of per out image.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>param</i>	The information of per affine transformation.
<i>stream</i>	CU kernel run in the stream.

1.1.4.2 void BatchRGBBilinearResizeNormPadPlane (uint8_t * *in_buf*, float * *out_buf*, int *batch*, int *input_batch_offset*, int *output_batch_offset*, batch_resize_param * *param*, float *scale*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *pad1*, float *pad2*, float *pad3*, bool *channel_rev*, cudaStream_t *stream*)

Multiple batch resize RGB/BGR image with bilinear interpolation and normalization.

Output format is RRRGGGBBB / BBBGGGRRR. Normalization function is: output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_out * h_out * 3) * sizeof(float).
<i>batch</i>	Number of input images.
<i>input_batch_offset</i>	The stride of input per batch.

Parameters

<i>output_batch_offset</i>	The stride of output per batch.
<i>param</i>	The information of per batch.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>channel_rev</i>	True mean rgb convert to bgr or bgr convert to rgb. False mean input and output image is same format.
<i>stream</i>	CU kernel run in the stream.

1.1.4.3 void FullRangeNV12ToRGBBilinearResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

NV12 full range(0~255) image convert to RGB and resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3 / 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_h * 3) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.4 void FullRangeYU12ToRGBBilinearResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

YU12(i420) full range(0~255) image convert to RGB and resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_h * 3) * sizeof(float)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.5 void GrayAffine (uint8_t * *input*, uint8_t * *output*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *out_w*, int *out_h*, float *m*[6], cudaStream_t *stream*)

Affine transformation gray image with bilinear interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_h) * sizeof(uint8_t)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>m</i> [6]	Affine transformation matrix.
<i>stream</i>	CU kernel run in the stream.

1.1.4.6 void GrayResizeBilinear (uint8_t * *in_buf*, uint8_t * *out_buf*, int *w_in*, int *h_in*, int *w_out*, int *h_out*, cudaStream_t *stream*)

Resize gray image with bilinear interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(w_in * h_in * 3) * sizeof(uint8_t)$.
---------------	--

Parameters

<i>out_buf</i>	The output buffer allocate in device memory. The size is $(w_out * h_out * 3) * \text{sizeof}(\text{uint8_t})$.
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.7 void GrayResizeNearest (uint8_t * *in_buf*, uint8_t * *out_buf*, int *in_w*, int *in_h*, int *out_w*, int *out_h*, cudaStream_t *stream*)

Resize gray image with nearest neighbor interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_h * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.8 void MultiRoiRGBBilinearResizeNormPadPlane (uint8_t * *in_buf*, float * *out_buf*, int *batch*, int *in_h*, int *in_w*, int *output_batch_offset*, multi_roi_resize_param * *param*, float *scale*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *pad1*, float *pad2*, float *pad3*, bool *channel_rev*, cudaStream_t *stream*)

Multiple roi resize RGB/BGR image ROI area with bilinear interpolation and normalization.

Output format is RRRGGGBBB / BBBGGGRRR. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(w_in * h_in * 3) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(w_out * h_out * 3) * \text{sizeof}(\text{float})$.
<i>batch</i>	Number of input images.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>output_batch_offset</i>	The stride of output per batch.
<i>param</i>	The information of per batch.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>channel_rev</i>	True mean rgb convert to bgr or bgr convert to rgb. False mean input and output image is same format.
<i>stream</i>	CU kernel run in the stream.

1.1.4.9 void non_max_suppression (float * *predict*, int *num_batch*, int *num_bboxes*, int *num_classes*, float *confidence_threshold*, float *nms_threshold*, float * *pout*, int *max_objects*, cudaStream_t *stream*)

Non max suppression inner class.

The num_bboxes need has same value between batches. At each one batch, the predict data and out data: predict size: num_bboxes * (5 + num_classes); predict item: [[cx, cy, w, h, obj_conf, cls0_conf, ..., clsN_conf] ... [cx, cy, w, h, obj_conf, cls0_conf, ..., clsN_conf]]; pout size: 1 + max_objects * 7 pout item: [box_cnt, [x1, y1, x2, y2, confidence, class_id, keep_flag], ... [x1, y1, x2, y2, confidence, keep_flag]]

Parameters

<i>predict</i>	The multi-batches input buffer allocate in device memory.
<i>num_batch</i>	The number of batch.
<i>num_classes</i>	The number of class.
<i>confidence_threshold</i>	The confidence threshold used for filter out boxes.
<i>nms_threshold</i>	The nms threshold used for merge boxes.
<i>pout</i>	The multi-batches output buffer allocate in device memory.
<i>max_objects</i>	The max boxes to keep
<i>stream</i>	The CU kernel run in the stream.

1.1.4.10 void NV12Crop (uint8_t * *in_buf*, uint8_t * *out_buf*, int *start_w*, int *start_h*, int *w_in*, int *h_in*, int *w_out*, int *h_out*, cudaStream_t *stream*)

Crop NV12 image.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3 / 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_out * h_out * 3 / 2) * sizeof(uint8_t).
<i>start_w</i>	The start index of crop area in the width dimension of input image.
<i>start_h</i>	The start index of crop area in the height dimension of input image.
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.11 void NV12ToBGRBilinearResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

NV12 image convert to BGR and resize with padding and normalization.

The order of BGR image is BBBGGGRRRR. Normalization function is: output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3 / 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_h * 3) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.12 void NV12ToBGRNearestResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

NV12 image convert to BGR and resize with padding and normalization.

The order of BGR image is BBBGGGRRRR. Normalization function is: output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3 / 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_h * 3) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.13 void NV12ToRGBBilinearResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

NV12 image convert to RGB and resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_h} * 3) * \text{sizeof}(\text{float})$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.14 void NV12ToRGBNearestResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

NV12 image convert to RGB and resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_h} * 3) * \text{sizeof}(\text{float})$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.

Parameters

<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.15 `template<typename DataType , typename IndxType , bool IsAscending = false> int32_t RadixSortFunc (DataType * data_dev, IndxType * indxs_dev, int32_t total_num, DataType * data_out_dev = nullptr, IndxType * indxs_out_dev = nullptr, void * workspace_dev = nullptr, cudaStream_t * stream = nullptr)`

Sort data with radix sort algorithm.

Parameters

<i>data_dev</i>	The keys for sort.
<i>indxs_dev</i>	The indexes which pair with key. Calculate by call this function with nullptr of data_dev, indxs_dev, data_out_dev and indxs_out_dev.
<i>total_num</i>	The total number of keys need to be sort.
<i>data_out_dev</i>	The sorted keys.
<i>data_out_dev</i>	The sorted indexes.
<i>workspace_dev</i>	The workspace buffer allocate in device memory.
<i>stream</i>	CU kernel run in the stream.

1.1.4.16 `void RGB2YU12 (uint8_t * in_buf, uint8_t * out_buf, int w, int h, cudaStream_t stream)`

RGB image convert to YUV420p(I420).

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_w * 3 / 2) * sizeof(uint8_t).
<i>w</i>	The width of input image.
<i>h</i>	The height of input image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.17 `void RGBAffine (uint8_t * input, uint8_t * output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], cudaStream_t stream)`

Affine transformation rgb image with bilinear interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_h * 3) * sizeof(uint8_t).

Parameters

<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>m[6]</i>	Affine transformation matrix.
<i>stream</i>	CU kernel run in the stream.

1.1.4.18 void RGBBilinearResizeCropNormPlaneV2 (uint8_t * *in_buf*, float * *out_buf*, uchar4 * *ws*, int *in_w*, int *in_h*, int *resized_w*, int *resized_h*, int *crop_w_start*, int *crop_h_start*, int *crop_w*, int *crop_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, bool *fmt_cvt*, cudaStream_t *stream*)

RGB resize(compatible with transforms.Resize of torchvision) with crop and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output[channel]} = (\text{input[channel]} * \text{scale} - \text{mean[channel]}) * \text{standard[channel]}$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_h} * 3) * \text{sizeof}(\text{float})$.
<i>ws</i>	The workspace buffer allocate in device memory.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>resized_w</i>	The width of resized image.
<i>resized_h</i>	The height of resized image.
<i>crop_w_start</i>	The start index of crop area in the width dimension of resized image.
<i>crop_h_start</i>	The start index of crop area in the height dimension of resized image.
<i>crop_w</i>	The width of output image.
<i>crop_h</i>	The height of output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>fmt_cvt</i>	Convert rgb to bgr or not.
<i>stream</i>	CU kernel run in the stream.

1.1.4.19 void RGBCrop (uint8_t * *in_buf*, uint8_t * *out_buf*, int *start_w*, int *start_h*, int *w_in*, int *h_in*, int *w_out*, int *h_out*, cudaStream_t *stream*)

Crop RGB image.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_out * h_out * 3) * sizeof(uint8_t).
<i>start↔ _w</i>	The start index of crop area in the width dimension of input image.
<i>start↔ _h</i>	The start index of crop area in the height dimension of input image.
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.20 void RGBNormalization (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *in_c*, float *mean*, float *standard*, float *scale*, cudaStream_t *stream*)

RGB image normalization.

output = (input * scale - mean) * standard

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * in_c) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (in_w * in_h * in_c) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>in_c</i>	The channels of input image.
<i>mean,standard,scale</i>	The parameter in "output = (input * scale - mean) * standard"
<i>stream</i>	CU kernel run in the stream.

1.1.4.21 void RGBNormalization_3Channels (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, float *mean1*, float *mean2*, float *mean3*, float *standard1*, float *standard2*, float *standard3*, float *scale*, bool *input_plane*, bool *output_plane*, bool *channel_rev*, cudaStream_t *stream*)

RGB image normalization.

output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (in_w * in_h * 3) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.

Parameters

<i>mean1,mean2,mean3,standard1,standard2,standard3,scale</i>	The parameter in "output[channel_out] = (input[channel] * scale - mean[channel]) * standard[channel]".
<i>input_plane</i>	True mean input image is RRRGGG BBB. False mean input image is RGBRGBRGB.
<i>output_plane</i>	True mean output image is RRRGGG BBB. False mean output image is RGBRGBRGB.
<i>channel_rev</i>	True mean rgb convert to bgr or bgr convert to rgb. False mean input and output image is same format.
<i>stream</i>	CU kernel run in the stream.

1.1.4.22 `void RGBResizeBilinear (uint8_t * in_buf, uint8_t * out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)`

Resize RGB image with bilinear interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_out * h_out * 3) * sizeof(uint8_t).
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.23 `void RGBResizeNearest (uint8_t * in_buf, uint8_t * out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)`

Resize RGB image with nearest neighbor interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_out * h_out * 3) * sizeof(uint8_t).
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.24 `void RGBResizePlaneBilinear (uint8_t * in_buf, uint8_t * out_buf, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)`

Resize RGB image with bilinear interpolation.

The order of RGB image is RRRGGGBBB.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(w_in * h_in * 3) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(w_out * h_out * 3) * \text{sizeof}(\text{uint8_t})$.
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

```
1.1.4.25 void RGBResizePlaneNearest ( uint8_t * in_buf, uint8_t * out_buf, int w_in, int h_in, int w_out, int h_out,
                                     cudaStream_t stream )
```

Resize RGB image with nearest interpolation.

The order of RGB image is RRRGGGBBB.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(w_in * h_in * 3) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(w_out * h_out * 3) * \text{sizeof}(\text{uint8_t})$.
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

```
1.1.4.26 void RGBResizePlanePadBilinear ( uint8_t * in_buf, uint8_t * out_buf, int w_in, int h_in, int w_out, int h_out, int
                                         w_box, int h_box, int w_b, int h_b, cudaStream_t stream )
```

Resize RGB image with bilinear interpolation.

The order of input and output RGB image is RRRGGGBBB. Output image has padding.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(w_in * h_in * 3) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(w_box * h_box * 3) * \text{sizeof}(\text{uint8_t})$.
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of resized image.
<i>h_out</i>	The height of resized image.
<i>w_box</i>	The width of output image.
<i>h_box</i>	The height of output image.
<i>w_b</i>	The offset of width dimension of resized image in output image.
<i>h_b</i>	The offset of height dimension of resized image in output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.27 void RGBResizePlanePadNearest (uint8_t * *in_buf*, uint8_t * *out_buf*, int *w_in*, int *h_in*, int *w_out*, int *h_out*, int *w_box*, int *h_box*, int *w_b*, int *h_b*, cudaStream_t *stream*)

Resize RGB image with nearest interpolation.

The order of input and output RGB image is RRRGGGBBB. Output image has padding.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_box * h_box * 3) * sizeof(uint8_t).
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of resized image.
<i>h_out</i>	The height of resized image.
<i>w_box</i>	The width of output image.
<i>h_box</i>	The height of output image.
<i>w_b</i>	The offset of width dimension of resized image in output image.
<i>h_b</i>	The offset of height dimension of resized image in output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.28 void RGBResizeWithROIBilinear (uint8_t * *in_buf*, uint8_t * *out_buf*, int *w_in*, int *h_in*, int *w_out*, int *h_out*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, cudaStream_t *stream*)

Resize RGB image ROI area with bilinear interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_out * h_out * 3) * sizeof(uint8_t).
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.29 void RGBResizeWithROINearest (uint8_t * *in_buf*, uint8_t * *out_buf*, int *w_in*, int *h_in*, int *w_out*, int *h_out*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, cudaStream_t *stream*)

Resize RGB image ROI area with nearest interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_out * h_out * 3) * sizeof(uint8_t).
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.30 void RGBROIBilinearResizeNormPadPlane (uint8_t * *in_buf*, float * *out_buf*, int *w_in*, int *h_in*, int *w_out*, int *h_out*, int *img_w*, int *img_h*, int *pad_w*, int *pad_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, float *scale*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *pad1*, float *pad2*, float *pad3*, bool *channel_rev*, cudaStream_t *stream*)

Resize RGB/BGR image ROI area with bilinear interpolation and normalization.

Output format is RRRGGG BBBB / BBBGGG RRRR. Normalization function is: output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_out * h_out * 3) * sizeof(float).
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>channel_rev</i>	True mean rgb convert to bgr or bgr convert to rgb. False mean input and output image is same format.
<i>stream</i>	CU kernel run in the stream.

1.1.4.31 void RGBROIBilinearResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *w_in*, int *h_in*, int *w_out*, int *h_out*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, float *scale*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, bool *channel_rev*, cudaStream_t *stream*)

Resize RGB/BGR image ROI area with bilinear interpolation and normalization.

Output format is RRRGGG BBBB / BBBGGG RRRR. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_out * h_out * 3) * sizeof(float).
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>mean1, mean2, mean3, std1, std2, std3, scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>channel_rev</i>	True mean rgb convert to bgr or bgr convert to rgb. False mean input and output image is same format.
<i>stream</i>	CU kernel run in the stream.

1.1.4.32 void RGBROINearestResizeNormPadPlane (uint8_t * *in_buf*, float * *out_buf*, int *w_in*, int *h_in*, int *w_out*, int *h_out*, int *img_w*, int *img_h*, int *pad_w*, int *pad_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, float *scale*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *pad1*, float *pad2*, float *pad3*, bool *channel_rev*, cudaStream_t *stream*)

Resize RGB/BGR image ROI area with nearest interpolation and normalization.

Output format is RRRGGG BBBB / BBBGGG RRRR. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_out * h_out * 3) * sizeof(float).
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.

Parameters

<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>channel_rev</i>	True mean rgb convert to bgr or bgr convert to rgb. False mean input and output image is same format.
<i>stream</i>	CU kernel run in the stream.

1.1.4.33 void RGBROINearestResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *w_in*, int *h_in*, int *w_out*, int *h_out*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, float *scale*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, bool *channel_rev*, cudaStream_t *stream*)

Resize RGB/BGR image ROI area with nearest interpolation and normalization.

Output format is RRRGGG BBBB / BBBGGG RRRR. Normalization function is: output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (w_in * h_in * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (w_out * h_out * 3) * sizeof(float).
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>channel_rev</i>	True mean rgb convert to bgr or bgr convert to rgb. False mean input and output image is same format.
<i>stream</i>	CU kernel run in the stream.

1.1.4.34 void RGBRotate (uint8_t * *in*, uint8_t * *out*, int *in_w*, int *in_h*, int *rotate_mode*, cudaStream_t *stream*)

Rotate RGB image with specific rotate mode.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_w * 3) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>rotate_mode</i>	The rotate angle of clockwise which include 90 degree, 180 degree, 270 degree.
<i>stream</i>	CU kernel run in the stream.

1.1.4.35 void RoiNv122RGBAffineNorm (uint8_t * *input*, float * *output*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *out_w*, int *out_h*, float *m*[6], float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

NV12 image roi area convert to RGB and affine transformation with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3 / 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_w * 3) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>m</i> [6]	Affine transformation matrix.
<i>mean1, mean2, mean3, std1, std2, std3, scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1, pad2, pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.36 void RoiNV12ToBGRBilinearResizePlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

NV12 image roi area convert to BGR and resize with padding.

The order of BGR image is BBBGGGRRR.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_w * 3) * sizeof(float)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.37 void RoiNV12ToRGBBilinearResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

NV12 image roi area convert to RGB and resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_w * 3) * sizeof(float)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.

Parameters

<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.38 void RoiNV12ToRGBBilinearResizePlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

NV12 image roi area convert to RGB and resize with padding.

The order of RGB image is RRRGGGBBB.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3 / 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_w * 3) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.39 void RoiNV12ToRGBBilinearResizeQuantizePlane (uint8_t * *in_buf*, uint8_t * *out_buf*, uchar4 * *ws*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *pad1*, float *pad2*, float *pad3*, float *zero_point*, float *scales_input*, cudaStream_t *stream*)

NV12 image roi area convert to RGB, bilinear resize, padding and quantize output to uint8.

The order of RGB image is RRRGGGBBB. Quantize function is: output[channel] = (input * scales_input + zero_point

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3 / 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_h * 3) * sizeof(uint8_t).

Parameters

<i>ws</i>	The workspace buffer allocate in device memory. The size is (roi_w * roi_h) * sizeof(uchar4).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>zero_point</i>	The output quantize parameter.
<i>scales_input</i>	The output quantize parameter.
<i>stream</i>	CU kernel run in the stream.

1.1.4.40 void RoiYU12BGRAffineNorm (uint8_t * *input*, float * *output*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *out_w*, int *out_h*, float *m*[6], float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

YU12 image roi area convert to BGR and affine transformation with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3 / 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_h * 3) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>m</i> [6]	Affine transformation matrix.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.41 `void RoiYU122RGBAffineNorm (uint8_t * input, float * output, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int out_w, int out_h, float m[6], float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)`

YU12 image roi area convert to RGB and affine transformation with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_h} * 3) * \text{sizeof}(\text{float})$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>m[6]</i>	Affine transformation matrix.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.42 `void RoiYU12ToBGRBilinearResizePlane (uint8_t * in_buf, float * out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float pad1, float pad2, float pad3, cudaStream_t stream)`

YU12(i420) image roi area convert to BGR and resize with padding and normalization.

The order of BGR image is BBBGGGRRR. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_h} * 3) * \text{sizeof}(\text{float})$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.

Parameters

<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.43 `void RoiYU12ToRGBBilinearResizeNormPlane (uint8_t * in_buf, float * out_buf, int in_w, int in_h, int roi_w_start, int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)`

YU12(i420) image roi area convert to RGB and resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output[channel]} = (\text{input[channel]} * \text{scale} - \text{mean[channel]}) * \text{standard[channel]}$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_h} * 3) * \text{sizeof}(\text{float})$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.44 void RoiYU12ToRGBBilinearResizeNormPlaneV2 (uint8_t * *in_buf*, float * *out_buf*, uchar4 * *ws*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *out_w*, int *out_h*, int *input_pad_w*, int *input_pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, uint8_t *pad1*, uint8_t *pad2*, uint8_t *pad3*, cudaStream_t *stream*)

YU12(i420) image roi area convert to RGB and resize(compatible with transforms.Resize of torchvision) with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_w} * 3) * \text{sizeof}(\text{float})$.
<i>ws</i>	The workspace buffer allocate in device memory.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>input_pad_w</i>	The pad of width dimension of input image.
<i>input_pad_h</i>	The pad of height dimension of input image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in " $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{std}[\text{channel}]$ ".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.45 void RoiYU12ToRGBBilinearResizePlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

YU12(i420) image roi area convert to RGB and resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_w} * 3) * \text{sizeof}(\text{float})$.
<i>in_w</i>	The width of input image.

Parameters

<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.46 void RoiYU12ToRGBBilinearResizeQuantizePlane (uint8_t * *in_buf*, uint8_t * *out_buf*, uchar4 * *ws*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *pad1*, float *pad2*, float *pad3*, float *zero_point*, float *scales_input*, cudaStream_t *stream*)

YU12 image roi area convert to RGB, bilinear resize, padding and quantize output to uint8.

The order of RGB image is RRRGGGBBB. Quantize function is: output[channel] = (input * scales_input + zero_point

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3 / 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_h * 3) * sizeof(uint8_t).
<i>ws</i>	The workspace buffer allocate in device memory. The size is (roi_w * roi_h) * sizeof(uchar4).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>zero_point</i>	The output quantize parameter.
<i>scales_input</i>	The output quantize parameter.
<i>stream</i>	CU kernel run in the stream.

1.1.4.47 void RoiYUV400PToRGBBilinearResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean*, float *std*, float *scale*, float *pad*, cudaStream_t *stream*)

YUV400P image roi area convert to RGB and bilinear resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_h) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean,std,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.48 void RoiYUV422PToRGBBilinearResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

YUV422P image roi area convert to RGB and bilinear resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_h * 3) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.

Parameters

<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

```
1.1.4.49 void RoiYUV422ToRGBBilinearResizeNormPlane ( uint8_t * in_buf, float * out_buf, int in_w, int in_h, int roi_w_start,
int roi_h_start, int roi_w, int roi_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float
mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t
stream )
```

YUV422 image roi area convert to RGB and bilinear resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_h * 3) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.50 void RoiYUV444PToRGBBilinearResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *roi_w_start*, int *roi_h_start*, int *roi_w*, int *roi_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

YUV444P image roi area convert to RGB and bilinear resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_h} * 3) * \text{sizeof}(\text{float})$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>roi_w_start</i>	The start index of ROI area in the width dimension of input image.
<i>roi_h_start</i>	The start index of ROI area in the height dimension of input image.
<i>roi_w</i>	The width of ROI area in the width dimension of input image.
<i>roi_h</i>	The height of ROI area in the height dimension of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.51 void YU122RGBAffine (uint8_t * *input*, uint8_t * *output*, int *in_w*, int *in_h*, int *out_w*, int *out_h*, float *m[6]*, cudaStream_t *stream*)

YU12(i420) image convert to RGB and affine transformation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_h} * 3) * \text{sizeof}(\text{uint8_t})$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>m[6]</i>	Affine transformation matrix.
<i>stream</i>	CU kernel run in the stream.

1.1.4.52 `void YU12Crop (uint8_t * in_buf, uint8_t * out_buf, int start_w, int start_h, int w_in, int h_in, int w_out, int h_out, cudaStream_t stream)`

Crop YU12 image.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(w_in * h_in * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(w_out * h_out * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>start_w</i>	The start index of crop area in the width dimension of input image.
<i>start_h</i>	The start index of crop area in the height dimension of input image.
<i>w_in</i>	The width of input image.
<i>h_in</i>	The height of input image.
<i>w_out</i>	The width of output image.
<i>h_out</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.53 `void YU12ToBGRBilinearResizeNormPlane (uint8_t * in_buf, float * out_buf, int in_w, int in_h, int img_w, int img_h, int out_w, int out_h, int pad_w, int pad_h, float mean1, float mean2, float mean3, float std1, float std2, float std3, float scale, float pad1, float pad2, float pad3, cudaStream_t stream)`

YU12(i420) image convert to BGR and resize with padding and normalization.

The order of BGR image is BBBGGGRRRR. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_h * 3) * \text{sizeof}(\text{float})$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.54 void YU12ToBGRNearestResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

YU12(i420) image convert to BGR and resize with padding and normalization.

The order of BGR image is BBBGGGRRR. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_h} * 3) * \text{sizeof}(\text{float})$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.55 void YU12ToRGBBilinearResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

YU12(i420) image convert to RGB and resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: $\text{output}[\text{channel}] = (\text{input}[\text{channel}] * \text{scale} - \text{mean}[\text{channel}]) * \text{standard}[\text{channel}]$

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(\text{in_w} * \text{in_h} * 3 / 2) * \text{sizeof}(\text{uint8_t})$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(\text{out_w} * \text{out_h} * 3) * \text{sizeof}(\text{float})$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.

Parameters

<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.56 void YU12ToRGBNearestResizeNormPlane (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, int *img_w*, int *img_h*, int *out_w*, int *out_h*, int *pad_w*, int *pad_h*, float *mean1*, float *mean2*, float *mean3*, float *std1*, float *std2*, float *std3*, float *scale*, float *pad1*, float *pad2*, float *pad3*, cudaStream_t *stream*)

YU12(i420) image convert to RGB and resize with padding and normalization.

The order of RGB image is RRRGGGBBB. Normalization function is: output[channel] = (input[channel] * scale - mean[channel]) * standard[channel]

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is (in_w * in_h * 3 / 2) * sizeof(uint8_t).
<i>out_buf</i>	The output buffer allocate in device memory. The size is (out_w * out_h * 3) * sizeof(float).
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>img_w</i>	The width of resized image.
<i>img_h</i>	The height of resized image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>pad_w</i>	The offset of width dimension of resized image in output image.
<i>pad_h</i>	The offset of height dimension of resized image in output image.
<i>mean1,mean2,mean3,std1,std2,std3,scale</i>	The parameter in "output[channel] = (input[channel] * scale - mean[channel]) * std[channel]".
<i>pad1,pad2,pad3</i>	The padding value of output.
<i>stream</i>	CU kernel run in the stream.

1.1.4.57 void YUV440pToYUV420p (uint8_t * *sy*, uint8_t * *su*, uint8_t * *sv*, uint8_t * *dy*, uint8_t * *du*, uint8_t * *dv*, int *w*, int *h*, int *align_w*, cudaStream_t *stream*)

yuv440p transform to yuv420p.

Parameters

<i>sy,su,sv</i>	The input buffer of [Y, U, V] 3 channels allocate in device memory.
<i>dy,du,dv</i>	The output buffer of [Y, U, V] 3 channels allocate in device memory.
<i>w</i>	The width of image.
<i>h</i>	The height of image.
<i>align_w</i>	The stride of height of input image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.58 void YUV444pToYUV420p (uint8_t * *sy*, uint8_t * *su*, uint8_t * *sv*, uint8_t * *dy*, uint8_t * *du*, uint8_t * *dv*, int *w*, int *h*, int *align_w*, cudaStream_t *stream*)

yuv444p transform to yuv420p.

Parameters

<i>sy,su,sv</i>	The input buffer of [Y, U, V] 3 channels allocate in device memory.
<i>dy,du,dv</i>	The output buffer of [Y, U, V] 3 channels allocate in device memory.
<i>w</i>	The width of image.
<i>h</i>	The height of image.
<i>align_w</i>	The stride of height of input image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.59 void YUVI420ResizeBilinear (uint8_t * *in_buf*, uint8_t * *out_buf*, int *in_w*, int *in_h*, int *out_w*, int *out_h*, cudaStream_t *stream*)

Resize YUV(i420) image with bilinear interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_h * 3 / 2) * sizeof(uint8_t)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.60 void YUVI420ResizeNearest (uint8_t * *in_buf*, uint8_t * *out_buf*, int *in_w*, int *in_h*, int *out_w*, int *out_h*, cudaStream_t *stream*)

Resize YUV(i420) image with nearest neighbor interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_h * 3 / 2) * sizeof(uint8_t)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.61 void YUVNv12ResizeBilinear (uint8_t * *in_buf*, uint8_t * *out_buf*, int *in_w*, int *in_h*, int *out_w*, int *out_h*, cudaStream_t *stream*)

Resize YUV(nv12) image with bilinear interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_h * 3 / 2) * sizeof(uint8_t)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.62 void YUVNv12ResizeNearest (uint8_t * *in_buf*, uint8_t * *out_buf*, int *in_w*, int *in_h*, int *out_w*, int *out_h*, cudaStream_t *stream*)

Resize YUV(nv12) image with nearest neighbor interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_h * 3 / 2) * sizeof(uint8_t)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.63 void YUVNv12ToRGB (uint8_t * *in_buf*, uint8_t * *out_buf*, int *in_w*, int *in_h*, cudaStream_t *stream*)

Convert YUV(nv12) image to RGB image.

The order of RGB image is RGBRGBRGB.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(in_w * in_h * 3) * sizeof(uint8_t)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.64 void YUVNv12ToRGBFloat (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, cudaStream_t *stream*)

Convert YUV(nv12) image to RGB image.

The order of RGB image is RGBRGBRGB.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(in_w * in_h * 3) * sizeof(float)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.65 void YUVNv12ToRGBPlane (uint8_t * *in_buf*, uint8_t * *out_buf*, int *in_w*, int *in_h*, cudaStream_t *stream*)

Convert YUV(nv12) image to RGB image.

The order of RGB image is RRRGGGBBB.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(in_w * in_h * 3) * sizeof(uint8_t)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.66 void YUVNv21ResizeBilinear (uint8_t * *in_buf*, uint8_t * *out_buf*, int *in_w*, int *in_h*, int *out_w*, int *out_h*, cudaStream_t *stream*)

Resize YUV(nv21) image with bilinear interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_h * 3 / 2) * sizeof(uint8_t)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.67 void YUVNv21ResizeNearest (uint8_t * *in_buf*, uint8_t * *out_buf*, int *in_w*, int *in_h*, int *out_w*, int *out_h*, cudaStream_t *stream*)

Resize YUV(nv21) image with nearest neighbor interpolation.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(out_w * out_h * 3 / 2) * sizeof(uint8_t)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>out_w</i>	The width of output image.
<i>out_h</i>	The height of output image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.68 void YUVYu12ToRGB (uint8_t * *in_buf*, uint8_t * *out_buf*, int *in_w*, int *in_h*, cudaStream_t *stream*)

Convert YUV(yu12, also called i420) image to RGB image.

The order of RGB image is RGBRGBRGB.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(in_w * in_h * 3) * sizeof(uint8_t)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.69 void YUVYu12ToRGBFloat (uint8_t * *in_buf*, float * *out_buf*, int *in_w*, int *in_h*, cudaStream_t *stream*)

Convert YUV(yu12, also called i420) image to RGB image.

The order of RGB image is RGBRGBRGB.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(in_w * in_h * 3) * sizeof(float)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>stream</i>	CU kernel run in the stream.

1.1.4.70 void YUVYu12ToRGBPlane (uint8_t * *in_buf*, uint8_t * *out_buf*, int *in_w*, int *in_h*, cudaStream_t *stream*)

Convert YUV(yu12, also called i420) image to RGB image.

The order of RGB image is RRRGGGBBB.

Parameters

<i>in_buf</i>	The input buffer allocate in device memory. The size is $(in_w * in_h * 3 / 2) * sizeof(uint8_t)$.
<i>out_buf</i>	The output buffer allocate in device memory. The size is $(in_w * in_h * 3) * sizeof(uint8_t)$.
<i>in_w</i>	The width of input image.
<i>in_h</i>	The height of input image.
<i>stream</i>	CU kernel run in the stream.

Index

- BatchRGBAffine
 - image_proc.h, [7](#)
- BatchRGBBilinearResizeNormPadPlane
 - image_proc.h, [7](#)
- FullRangeNV12ToRGBBilinearResizeNormPlane
 - image_proc.h, [8](#)
- FullRangeYU12ToRGBBilinearResizeNormPlane
 - image_proc.h, [8](#)
- GrayAffine
 - image_proc.h, [9](#)
- GrayResizeBilinear
 - image_proc.h, [9](#)
- GrayResizeNearest
 - image_proc.h, [10](#)
- image_proc.h, [1](#)
 - BatchRGBAffine, [7](#)
 - BatchRGBBilinearResizeNormPadPlane, [7](#)
 - FullRangeNV12ToRGBBilinearResizeNormPlane, [8](#)
 - FullRangeYU12ToRGBBilinearResizeNormPlane, [8](#)
 - GrayAffine, [9](#)
 - GrayResizeBilinear, [9](#)
 - GrayResizeNearest, [10](#)
 - MultiRoiRGBBilinearResizeNormPadPlane, [10](#)
 - NV12Crop, [11](#)
 - NV12ToBGRBilinearResizeNormPlane, [11](#)
 - NV12ToBGRNearestResizeNormPlane, [12](#)
 - NV12ToRGBBilinearResizeNormPlane, [12](#)
 - NV12ToRGBNearestResizeNormPlane, [13](#)
 - non_max_suppression, [11](#)
 - RGB2YU12, [14](#)
 - RGBAffine, [14](#)
 - RGBAffineInfo_t, [6](#)
 - RGBBilinearResizeCropNormPlaneV2, [15](#)
 - RGBCrop, [15](#)
 - RGBNormalization, [16](#)
 - RGBNormalization_3Channels, [16](#)
 - RGBROIBilinearResizeNormPadPlane, [20](#)
 - RGBROIBilinearResizeNormPlane, [21](#)
 - RGBROINearestResizeNormPadPlane, [21](#)
 - RGBROINearestResizeNormPlane, [22](#)
 - RGBResizeBilinear, [17](#)
 - RGBResizeNearest, [17](#)
 - RGBResizePlaneBilinear, [17](#)
 - RGBResizePlaneNearest, [18](#)
 - RGBResizePlanePadBilinear, [18](#)
 - RGBResizePlanePadNearest, [19](#)
 - RGBResizeWithROIBilinear, [19](#)
 - RGBResizeWithROINearest, [19](#)
 - RGBRotate, [22](#)
 - ROTATE_180_CLOCKWISE, [6](#)
 - ROTATE_270_CLOCKWISE, [6](#)
 - ROTATE_90_CLOCKWISE, [6](#)
 - RadixSortFunc, [14](#)
 - RoiNV12ToBGRBilinearResizePlane, [23](#)
 - RoiNV12ToRGBBilinearResizeNormPlane, [24](#)
 - RoiNV12ToRGBBilinearResizePlane, [25](#)
 - RoiNV12ToRGBBilinearResizeQuantizePlane, [25](#)
 - RoiNv122RGBAffineNorm, [23](#)
 - RoiYU122BGRAffineNorm, [26](#)
 - RoiYU122RGBAffineNorm, [27](#)
 - RoiYU12ToBGRBilinearResizePlane, [27](#)
 - RoiYU12ToRGBBilinearResizeNormPlane, [28](#)
 - RoiYU12ToRGBBilinearResizeNormPlaneV2, [28](#)
 - RoiYU12ToRGBBilinearResizePlane, [29](#)
 - RoiYU12ToRGBBilinearResizeQuantizePlane, [30](#)
 - RoiYUV400PToRGBBilinearResizeNormPlane, [30](#)
 - RoiYUV422PToRGBBilinearResizeNormPlane, [31](#)
 - RoiYUV422ToRGBBilinearResizeNormPlane, [32](#)
 - RoiYUV444PToRGBBilinearResizeNormPlane, [33](#)
 - YU122RGBAffine, [33](#)
 - YU12Crop, [34](#)
 - YU12ToBGRBilinearResizeNormPlane, [34](#)
 - YU12ToBGRNearestResizeNormPlane, [34](#)
 - YU12ToRGBBilinearResizeNormPlane, [35](#)
 - YU12ToRGBNearestResizeNormPlane, [36](#)
 - YUV440pToYUV420p, [36](#)
 - YUV444pToYUV420p, [37](#)
 - YUVI420ResizeBilinear, [37](#)
 - YUVI420ResizeNearest, [37](#)
 - YUVNv12ResizeBilinear, [37](#)
 - YUVNv12ResizeNearest, [38](#)
 - YUVNv12ToRGBFloat, [38](#)
 - YUVNv12ToRGBPlane, [39](#)
 - YUVNv12ToRGB, [38](#)
 - YUVNv21ResizeBilinear, [39](#)
 - YUVNv21ResizeNearest, [39](#)
 - YUVYu12ToRGBFloat, [40](#)
 - YUVYu12ToRGBPlane, [40](#)
 - YUVYu12ToRGB, [40](#)
- MultiRoiRGBBilinearResizeNormPadPlane
 - image_proc.h, [10](#)
- NV12Crop
 - image_proc.h, [11](#)

NV12ToBGRBilinearResizeNormPlane
 image_proc.h, [11](#)
 NV12ToBGRNearestResizeNormPlane
 image_proc.h, [12](#)
 NV12ToRGBBilinearResizeNormPlane
 image_proc.h, [12](#)
 NV12ToRGBNearestResizeNormPlane
 image_proc.h, [13](#)
 non_max_suppression
 image_proc.h, [11](#)

 RGB2YU12
 image_proc.h, [14](#)
 RGBAffine
 image_proc.h, [14](#)
 RGBAffineInfo_t
 image_proc.h, [6](#)
 RGBBilinearResizeCropNormPlaneV2
 image_proc.h, [15](#)
 RGBCrop
 image_proc.h, [15](#)
 RGBNormalization
 image_proc.h, [16](#)
 RGBNormalization_3Channels
 image_proc.h, [16](#)
 RGBROIBilinearResizeNormPadPlane
 image_proc.h, [20](#)
 RGBROIBilinearResizeNormPlane
 image_proc.h, [21](#)
 RGBROINearestResizeNormPadPlane
 image_proc.h, [21](#)
 RGBROINearestResizeNormPlane
 image_proc.h, [22](#)
 RGBResizeBilinear
 image_proc.h, [17](#)
 RGBResizeNearest
 image_proc.h, [17](#)
 RGBResizePlaneBilinear
 image_proc.h, [17](#)
 RGBResizePlaneNearest
 image_proc.h, [18](#)
 RGBResizePlanePadBilinear
 image_proc.h, [18](#)
 RGBResizePlanePadNearest
 image_proc.h, [19](#)
 RGBResizeWithROIBilinear
 image_proc.h, [19](#)
 RGBResizeWithROINearest
 image_proc.h, [19](#)
 RGBRotate
 image_proc.h, [22](#)
 ROTATE_180_CLOCKWISE
 image_proc.h, [6](#)
 ROTATE_270_CLOCKWISE
 image_proc.h, [6](#)
 ROTATE_90_CLOCKWISE
 image_proc.h, [6](#)
 RadixSortFunc
 image_proc.h, [14](#)

 RoiNV12ToBGRBilinearResizePlane
 image_proc.h, [23](#)
 RoiNV12ToRGBBilinearResizeNormPlane
 image_proc.h, [24](#)
 RoiNV12ToRGBBilinearResizePlane
 image_proc.h, [25](#)
 RoiNV12ToRGBBilinearResizeQuantizePlane
 image_proc.h, [25](#)
 RoiNv122RGBAffineNorm
 image_proc.h, [23](#)
 RoiYU122BGRAffineNorm
 image_proc.h, [26](#)
 RoiYU122RGBAffineNorm
 image_proc.h, [27](#)
 RoiYU12ToBGRBilinearResizePlane
 image_proc.h, [27](#)
 RoiYU12ToRGBBilinearResizeNormPlane
 image_proc.h, [28](#)
 RoiYU12ToRGBBilinearResizeNormPlaneV2
 image_proc.h, [28](#)
 RoiYU12ToRGBBilinearResizePlane
 image_proc.h, [29](#)
 RoiYU12ToRGBBilinearResizeQuantizePlane
 image_proc.h, [30](#)
 RoiYUV400PToRGBBilinearResizeNormPlane
 image_proc.h, [30](#)
 RoiYUV422PToRGBBilinearResizeNormPlane
 image_proc.h, [31](#)
 RoiYUV422ToRGBBilinearResizeNormPlane
 image_proc.h, [32](#)
 RoiYUV444PToRGBBilinearResizeNormPlane
 image_proc.h, [33](#)

 YU122RGBAffine
 image_proc.h, [33](#)
 YU12Crop
 image_proc.h, [34](#)
 YU12ToBGRBilinearResizeNormPlane
 image_proc.h, [34](#)
 YU12ToBGRNearestResizeNormPlane
 image_proc.h, [34](#)
 YU12ToRGBBilinearResizeNormPlane
 image_proc.h, [35](#)
 YU12ToRGBNearestResizeNormPlane
 image_proc.h, [36](#)
 YUV440pToYUV420p
 image_proc.h, [36](#)
 YUV444pToYUV420p
 image_proc.h, [37](#)
 YUVI420ResizeBilinear
 image_proc.h, [37](#)
 YUVI420ResizeNearest
 image_proc.h, [37](#)
 YUVNv12ResizeBilinear
 image_proc.h, [37](#)
 YUVNv12ResizeNearest
 image_proc.h, [38](#)
 YUVNv12ToRGBFloat
 image_proc.h, [38](#)

YUVNv12ToRGBPlane
 image_proc.h, [39](#)
YUVNv12ToRGB
 image_proc.h, [38](#)
YUVNv21ResizeBilinear
 image_proc.h, [39](#)
YUVNv21ResizeNearest
 image_proc.h, [39](#)
YUVYu12ToRGBFloat
 image_proc.h, [40](#)
YUVYu12ToRGBPlane
 image_proc.h, [40](#)
YUVYu12ToRGB
 image_proc.h, [40](#)