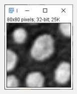
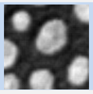
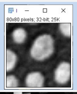
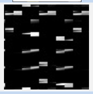
CLIJ cheat sheet: ImageJ macro I

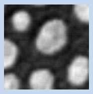
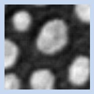
GPU-accelerated image processing in Fiji

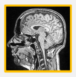
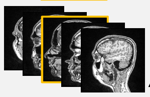
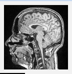
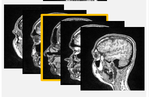
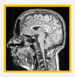
	Operation	Parameters	Result	Dim	Examples
Basics / Wrangling	Initialize CLIJ	[], HD, GFX or CPU			`run("CLIJ Macro Extensions", "cl_device=[]");`

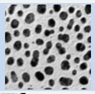
	Push			2D
				3D
					`// send current image to GPU`
					`input = getTitle();`
					`Ext.CLIJ_push(input);`

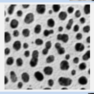
	Pull			2D
				3D
					`// get result image from GPU back`
					`Ext.CLIJ_pull(output);`

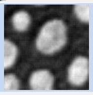
	Create	1024, 1024, 8		2D
				3D
					`Ext.CLIJ_create2D("new2D", w, h, bitDepth);`
					`Ext.CLIJ_create3D("new3D", w, h, depth, bitDepth);`

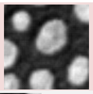
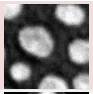
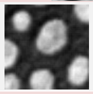
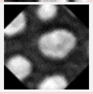
	Convert			2D
				3D
					`Ext.CLIJ_convertFloat(input, "result_float");`
					`Ext.CLIJ_convertUInt8(input, "result_uint8");`
					`Ext.CLIJ_convertUInt16(input, "result_uint16");`

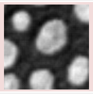
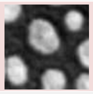
	Copy
	
	Copy slice
		
		 , 50
		 , 50
			
			
			
				2D
				3D
					`// duplicate`
					`Ext.CLIJ_copy(source, result);`
					
					`// put a slice into a stack`
					`Ext.CLIJ_copySlice(stack, slice, sliceIndex);`
					
					`// copy a slice out of a stack`
					`Ext.CLIJ_copySlice(slice, stack, sliceIndex);`

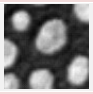
	Crop			2D
				3D
					`// crop image`
					`Ext.CLIJ_crop2D("original", "cropped", x, y, width, height);`

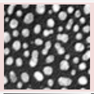
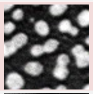
	Paste	 , 9, 9		2D
				3D
					`// paste image`
					`Ext.CLIJx_paste2D(“cropped”, “target”, x, y);`

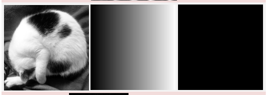
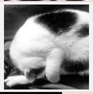
	Release				`// free / release memory occupied by an image`
					`Ext.CLIJ_release("image name");`

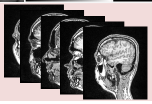
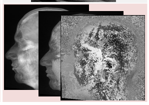
	Clear				`Ext.CLIJ_clear(); // empty GPU memory`
Spatial transforms	Rotate by 90 degrees			2D 3D	`Ext.CLIJ_rotateLeft(input, result);`
	Rotate	 , 45, true		2D 3D	`Ext.CLIJ_rotate2D(input, result, angle, rotateAroundCenter);`

	Flip	 , true, false		2D
				3D
					`Ext.CLIJ_flip2D(input, result, flipX, flipY);`
					`Ext.CLIJ_flip3D(input, result, flipX, flipY,flipZ);`

	Translate			2D 3D	`Ext.CLIJ_translate2D(input, result, shiftX, shiftY);`

	Affine transform			2D
				3D
					`transform = “center scale=2 rotate=45 -center”;`
					`Ext.CLIJ_affineTransform2D(source,result,transform);`

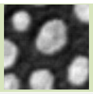
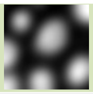
	Deform / warp			2D
				3D
					`// warp image`
					`Ext.CLIJ_applyVectorField2D(source, vectorFieldX, vectorFieldY, result);`

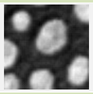
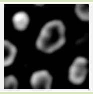
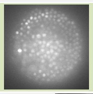
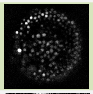
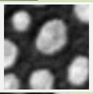
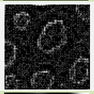
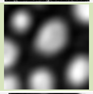
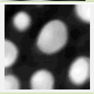
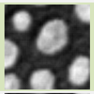
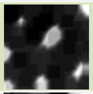
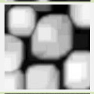
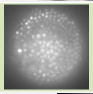
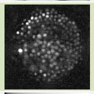
	Projections			3D
				->
				2D
					`Ext.CLIJ_argMaximumZProjection(in, result, arg_z);`
					`Ext.CLIJx_standardDeviationZProjection(in,result);`

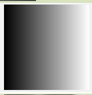
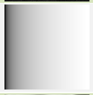
CLIJ cheat sheet: ImageJ macro II

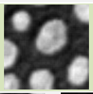
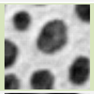
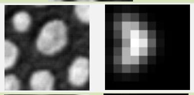
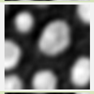
GPU-accelerated image processing in Fiji

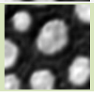
	Operation	Parameters	Result	Dim	Examples

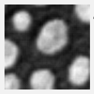
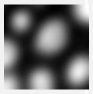
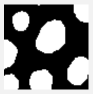
Filters	Gaussian blur	 , 10, 10		2D
				3D
					`Ext.CLIJ_blur2D(input, result, sigmaX, sigmaY);`
					`Ext.CLIJ_blur3D(input, result, sigmaX, sigmaY, sigmaZ);`

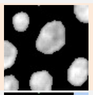
	Difference of Gaussian	 , 2, 2, 20, 20		2D 3D	`Ext.CLIJx_differenceOfGaussian2D(input, result, sigma1x, sigma1y, sigma2x, sigma2y);`
	Subtract background	 , 25, 25, 0		2D 3D	`Ext.CLIJx_subtractBackground3D(input, result, sigmaX, sigmaY, sigmaZ);`
	Laplace			2D 3D	`Ext.CLIJx_laplace(input, result);`
	Mean	, 5		2D 3D	`Ext.CLIJ_mean2DBox(input, result, radiusX, radiusY);`
	Median	, 5		2D 3D	`Ext.CLIJ_medianSliceBySliceBox(input, result, radiusX, radiusY);`
	Minimum	 , 5		2D 3D	`Ext.CLIJ_minimum2DBox(input, result, radiusX, radiusY);`
	Maximum	, 5		2D 3D	`Ext.CLIJ_maximum3DBox(input, result, radiusX, radiusY, radiusZ);`
	Top-hat	 , 25, 25, 0		2D 3D	`Ext.CLIJx_topHatBox(input, result, radiusX, radiusY, radiusZ);`

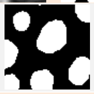
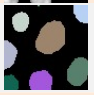
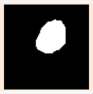
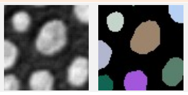
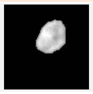
	Logarithm / Exponential			2D
				3D
					`Ext.CLIJx_logarithm(input, result);`
					`Ext.CLIJx_exponential(input, result);`

	Invert			2D 3D	`Ext.CLIJ_invert(input, result);`
	Convolve			2D 3D	`Ext.CLIJ_convolve(input, kernel, result);`

	Deconvolve			2D
				3D
					`// Richardson-Lucy decon.; for academic purposes`
					`Ext.CLIJ_deconvolve(input, kernel, result, iterations);`

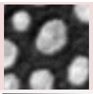
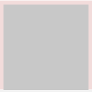
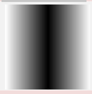
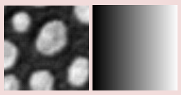
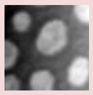
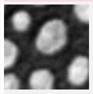
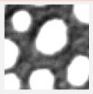
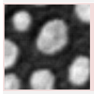
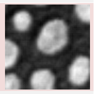
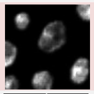
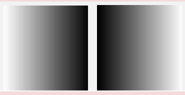
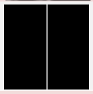
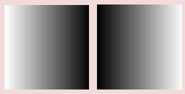
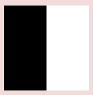
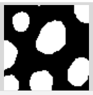
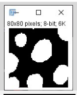
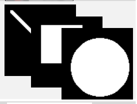
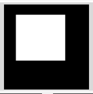
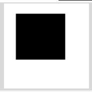
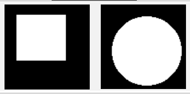
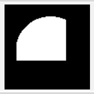
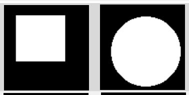
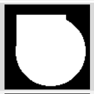
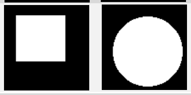
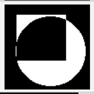
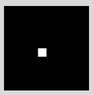
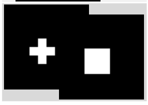
Segmentation / labeling	Threshold	 "Otsu", 127 or 		2D
				3D
					`Ext.CLIJ_threshold(input, binary_result, 127);`
					`Ext.CLIJ_automaticThreshold(input, binary_result, "Otsu");`
					`Ext.CLIJ_localThreshold(input, threshold_image, binary_result);`

	Mask			2D
				3D
					`// mask an image`
					`Ext.CLIJ_mask(input, mask, result);`

	Connected components			2D 3D	`Ext.CLIJx_connectedComponentsLabeling(binary_input, labelmap_result);`
	Label to mask	, 4		2D 3D	`Ext.CLIJx_labelToMask(labelmap_input, mask_result, label_index);`
	Mask labelled	 , 4		2D 3D	`Ext.CLIJx_maskLabel(input, labelmap, result, label_index);`

CLIJ cheat sheet: ImageJ macro III

GPU-accelerated image processing in Fiji

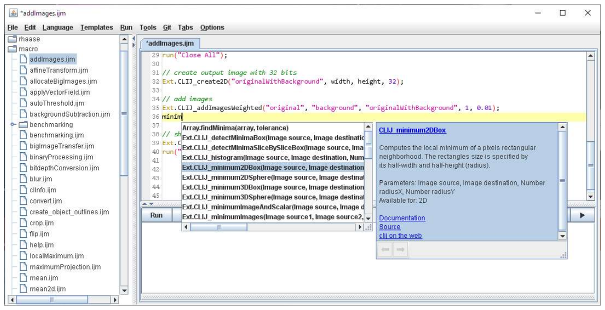
	Operation	Parameters	Result	Dim	Examples
Math	Set			2D 3D	`Ext.CLIJ_set(result, pixel_value);`
	Absolute |x|			2D 3D	`Ext.CLIJ_absolute(input, result);`
	Add / Subtract			2D 3D	`Ext.CLIJ_addImages(summand1, summand2, result);` `Ext.CLIJ_addImageAndScalar(input, result, scalar);` `Ext.CLIJ_addImagesWeighted(in1, in2, result, a,b);`
	Multiply / Divide	 , 2		2D 3D	`Ext.CLIJ_multiplyImages(input1, input2, result);` `Ext.CLIJ_multiplyImageAndScalar(input, result, n);` `Ext.CLIJ_divideImages(divident, divisor, result);`
	Power	 , 2 or 		2D 3D	`Ext.CLIJ_power(input, result, exponent);` `Ext.CLIJx_powerImages(input, exp_image, result);`
	Equal = Not Equal !=			2D 3D	`Ext.CLIJx_equal(source1, source2, result);` `Ext.CLIJx_notEqual(source1, source2, result);`
	Greater / Smaller			2D 3D	`Ext.CLIJx_greater(source1, source2, result);` `Ext.CLIJx_smaller(source1, source2, result);` `Ext.CLIJx_smallerOrEqual(source1, source2,result);`
Binary Images	PullBinary			2D 3D	`Ext.CLIJ_pullBinary(String image);`
	Draw line / box / sphere	10, 10, 50, 50		2D 3D	`Ext.CLIJx_drawLine(result, x1, y1, z1, x2, y2, z2, thickness);` `Ext.CLIJx_drawBox(result, x, y, z, width, height, depth);` `Ext.CLIJx_drawSphere(result, x, y, z, radius_x,radius_y,radius_z);`
	Not			2D 3D	`Ext.CLIJ_binaryNot(source, result);`
	And / Intersection			2D 3D	`Ext.CLIJ_binaryAnd(operand1, operand2, result);` `Ext.CLIJx_binaryInterection(op1, op2, result);`
	Or / Union			2D 3D	`Ext.CLIJ_binaryOr(operand1, operand2, result);` `Ext.CLIJ_binaryUnion(operand1, operand2, result);`
	XOr			2D 3D	`Ext.CLIJ_binaryXOr(operand1, operand2, result);`
	Dilate/ Erode			2D 3D	`Ext.CLIJ_dilateSphere(source, result);` `Ext.CLIJ_dilateBox(source, result);` `Ext.CLIJ_erodeSphereSliceBySlice(input, result);`

Detailed documentation

- The API reference is available online and is embedded in Fijis script editor. Start typing "CLIJ" in any macro and it will suggest CLIJ commands.
- Code examples are available on the CLIJ website

Installation instructions

- Install CLIJ by activating the "clij" update site in Fiji.
- Commands listed as "CLIJx" are under development and can be installed by adding the https://sites.imagej.net/clij2/ update site. Handle them with care.



```
run("Close All");

// create output image with 32 bits
Ext.CLIJ_create2D("originalWithBackground", width, height, 32);

// add images
Ext.CLIJ_addImagesWeighted("original", "background", "originalWithBackground", 1, 0.01);
minim

// sh
Ext.C
run("
```

```
Array.findMinima(array, tolerance)
Ext.CLIJ_detectMinimaBox(Image source, Image destinatio
Ext.CLIJ_detectMinimaSliceBySliceBox(Image source, Ima
Ext.CLIJ_histogram(Image source, Image destination, Num
Ext.CLIJ_minimum2DBox(Image source, Image destination
Ext.CLIJ_minimum2DSphere(Image source, Image destinat
Ext.CLIJ_minimum3DBox(Image source, Image destination
Ext.CLIJ_minimum3DSphere(Image source, Image destinat
Ext.CLIJ_minimumImageAndScalar(Image source, Image d
Ext.CLIJ_minimumImages(Image source1, Image source2,
```

CLIJ_minimum2DBox

Computes the local minimum of a pixels rectangular neighborhood. The rectangles size is specified by its half-width and half-height (radius).

Parameters: Image source, Image destination, Number radiusX, Number radiusY
Available for: 2D

Documentation
Source
clij on the web

https://clij.github.io/ @haesleinhuepf #clij cheat sheet 2019-09-06