Open CV Cheat Sheet

by thatguyandy27 via cheatography.com/42823/cs/12846/

Accessing and Modifying pixel values

Pixel value	`img[10 0,100]`
Accessing only blue pixel	`img[10 0,1 00,0]`
Modifying A Pixel	`img[10 0,100] = [255,2 55,255]`
Better pixel accessing	`img.it em( 10, 10,2)`
Better pixel modifying	`img.it ems et( (10 ,10 ,2` `,100)`
Access image properties	`img.shape`
Total number of pixels	`img.size`
Image datatype	`img.dtype`
Getting ROI	`ball = img[28 0:340, 330:390]`
Setting ROI	`img[27 3:333, 100:160] = ball`
Split Channels	`b,g,r = cv2.sp lit (img)` `b = img[:,:,0]`
Making Borders for Images	`cv2.co pyM ake Bor der (im g1, 10, 10, 10, 10, cv2.BO RDE R_R EPL ICATE)`

borderType	`cv2.BO RDE R_C ONSTANT`
	`cv2.BO RDE R_R EFLECT`
	`cv2.BO RDE R_R EFL ECT_101`
	`cv2.BO RDE R_R EPL ICATE`
	`cv2.BO RDE R_WRAP`

Arithmetic Operations on Images

Image Addition (OPENCV)	`print cv2.ad d(x,y) # 250+10 = 260 => 25 5`
Image Addition (Numpy)	`print x+y  # 250+10 = 260 % 256 = 4`

Arithmetic Operations on Images (cont)

Image Alpha Blending	`dst = cv2.ad dWe igh ted (im g1, 0.7, im g2, 0.3,0 )`
Bitwise AND	`img1_bg = cv2.bi twi se_ and (ro i,r oi,mask=mas k_inv)`
Bitwise NOT	`mask_inv = cv2.bi twi se_ not (mask)`

Morphological Transformations

Erosion	`erosion = cv2.er ode (im g,k ern el, ite rations = 1)`
Dilation	`dilation = cv2.di lat e(i mg, ker nel ,it era tions = 1)`
Opening	`opening = cv2.mo rph olo gyE x(img, cv2.MO RPH _OPEN, ke rnel)`
Closing	`closing = cv2.mo rph olo gyE x(img, cv2.MO RPH _CLOSE, k ernel)`
Morphological Gradient	`gradient = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)`
Top Hat	`tophat = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)`
Black Hat	`blackhat = cv2.morphologyEx(img, cv2.MORPH_BLACKHAT, kernel)`

Create Structuring Elements	`cv2.ge tSt ruc tur ing Ele men t(c v2.M OR PH_ REC T, ( 5 ,5))`
	`cv2.ge tSt ruc tur ing Ele men t(c v2.M OR PH_ ELL IPS E , ( 5,5))`
	`cv2.ge tSt ruc tur ing Ele men t(c v2.M OR PH_ CRO SS, ( 5,5))`

Performance Measurement and Improvement Techniques

Find # of clock--cycles	`e1 = cv2.ge tTi ckC oun`
	`# your code execution`
	`e2 = cv2.ge tTi ckC oun`
	`time = (e2 - e1)/ cv2.g`
	`cy()`

Find clock cycles per second	`cv2.ge tTi ckF req uenc`
Enable Optimizations	`cv2.se tUs eO p tim ize`
Measure Performance (IPython)	`%timeit y=x**2`
Performance Optimization Techniques	1. Avoid using loops in Python especially double/triple loops et slow. 2. Vectorize the algorithm/code possible extent because Numpy optimized for vector operations. 3. Exploit the cache coherence. 4. Never make copies of array to use views instead. Array cop operation.

Geometric Transformations of Images

Scaling Types	`cv2.IN TER _AREA`
	`cv2.IN TER _CUBIC`
	`cv2.IN TER _LINEAR`

Scaling	`res = cv2.re siz e(i mg,(2wi dth, 2heigh t), interp olation = cv2.IN TER _C UBIC)`
Shifting (100 x 50)	`M = np.flo at3 2([ [1, 0,1 00] ,[0 ,1, 50]])` `dst = cv2.wa rpA ffi ne( img ,M, (co ls, rows))`
Rotation	`M = cv2.ge tRo tat ion Mat rix 2D( (co ls/ 2,r ows /2) ,90,1)` `dst = cv2.wa rpA ffi ne( img ,M, (co ls, rows))`

Affine Transf-orm-ation	`pts1 = np.flo at3 2([ [50 ,50 ],[ 200 ,50 ],[ 50, 200]])`
	`pts2 = np.flo at3 2([ [10 ,10 0], [20 0,5 0], [10 0,2 50]])`
	`M = cv2.ge tAf fin eTr ans for m(p ts1 ,pts2)`
	`dst = cv2.wa rpA ffi ne( img ,M, (co ls, rows))`

Perspe-ctive Transf-orm-ation	`pts1 = np.flo at3 2([ [56 ,65 ],[ 368 ,52 ],[ 28, 387 ],[ 389, 390]])`
	`pts2 = np.flo at3 2([ [0, 0], [30 0,0 ],[ 0,3 00] ,[3 00, 300]])`
	`M = cv2.ge tPe rsp ect ive Tra nsf orm (pt s1, pts2)`
	`dst = cv2.wa rpP ers pec tiv (img, M, (300,300))`

Canny Edge Detection

Canny Detection	`edges = cv2.Ca nny (im g,1 00, 200)`

Image Pyramids

Lower Gaussian Pyramid	`lower_reso = cv2.py rDo wn( hig her_reso)`
Higher Gaussian Pyramid	`higher _reso2 = cv2.py rUp (lo wer_reso)`
Pyramid Blending	1. Load the two images 2. Find the Gaussian Pyramids 3. From Gaussian Pyramids, find their Laplacian Pyramids 4. Now each levels of Laplacian Pyramids 5. Finally from this joint image pyramids, recons-truct the original image

Changing Colorspaces

List Colorspace Flags (150+)	`flags = [i for i in dir(cv2) if i.star tsw ith ('C OLO R_')`
Convert to Gray	`img_gray = cv2.cv tCo lor (img, cv2. COLOR_ BGR 2GRAY)`
Convert to hsv	`hsv =  cv2.cv tCo lor (img, cv2. COLOR_ BGR 2HSV)`

Changing Colorspaces (cont)

Track Blue (color) Object	`lower_blue = np.arr ay( [11`
	`upper_blue = np.arr ay( [13`
	`mask = cv2.in Ran ge(hsv, l e)`
	`res = cv2.bi twi se_ and (f sk)`

Find HSV Color	`green = np.uin t8( [[[ 0,25` `hsv_green = cv2.cv tCo lor B GR2HSV)`

Image Thresholding

Thresh-olding Types	`cv2.TH RES H_B INARY`
	`cv2.TH RES H_B INA RY_INV`
	`cv2.TH RES H_TRUNC`
	`cv2.TH RES H_T OZERO`
	`cv2.TH RES H_T OZE RO_INV`

Getting Threshold	`ret,th resh4 = cv2.th res`
Adaptive Method Types	`cv2.AD APT IVE _TH RES H_` `cv2.AD APT IVE _TH RES H_`
Adaptive Threshold	`th3 = cv2.ad apt ive Thr ES H_B INA RY, 11,2)`
Otsu's Binari-zation	`ret3,th3 = cv2.th res hol`

Open CV Cheat Sheet

by thatguyandy27 via cheatography.com/42823/cs/12846/

Smoothing Images

Convolve an Image	`dst = cv2.fi lte r2D (im g,- 1,k ernel)`
Box (avera-ging) Filtering	`blur = cv2.bl ur( img ,(5,5))` `cv2.bo xFi lter()`
Create Gaussian Kernel	`cv2.ge tGa uss ian Ker nel (size, sigma, type)`
Gaussian Blur	`blur = cv2.Ga uss ian Blu r(i mg, (5, 5) ,0)`
Median Blur	`median = cv2.me dia nBl ur( img,5)`
Bilateral Blur	`blur = cv2.bi lat era lFi lte r(i mg, 9, 7 5,75)`

Image Gradients

Sobel	`sobelx = cv2.So bel (im g,c v2.C V_ 64F ,1, 0, k size=5)`
Laplacian	`laplacian = cv2.La pla cia n(i mg, cv2.CV _64F )`

*Output datatype cv2.CV_8U or np.uint8. So when you convert data to np.uint8, all negative slopes are made zero. In simple words, you miss that edge. If you want to detect both edges, better option is to keep the output datatype to some higher forms, like cv2.CV_16S, cv2.CV_64F etc, take its absolute value and then convert back to cv2.CV_8U

By **thatguyandy27**

cheatography.com/thatguyandy27/

Not published yet.
Last updated 17th September, 2017.