BCH2203 Python - 11. Computer Vision

Ramses van Zon

27 March 2024

Announcements

Next weeks lecture, which would be our last, will be postponed to the week after.

So no lecture next week.

A last assignment will be posted some time next week.

Computer Vision

. . . is focused on enabling machines to interpret and understand visual information from the real world.

In a biochemical context, this involves data from various forms of microscopy:

- Confocal Microscopy
- Fluorescence Microscopy
- Electron Microscopy
- Atomic Force Microscopy
- Fluorescence Resonance Energy Transfer
- Super-resolution microscopy

Beyond biochemistry, computer vision has healthcare, automotive, security, and entertainment applications.

What is OpenCV?

- OpenCV is an open-source computer vision and machine learning software library.
- It provides a wide range of functionalities for tasks such as:
  - image and video analysis
  - object detection and tracking
  - facial recognition, and more.
- OpenCV is written in C++ and has interfaces for C++, Python, Java, and MATLAB/Octave.
- Its Python interface is compatible with numpy and matplotlib.

Installation and Setup

On your own computer

```
$ pip install opencv-python  # consider a virtual env
```

or, which anaconda,

```
$ conda install opencv  # consider a conda env
```

On the Teach cluster

```
$ ssh -X lcl_uotphy1610sXXXX@teach.scinet.utoronto.ca
$ module load gcc/13 python/3.11 opencv/4.9.0
```

X forwarding: The `-X` option to ssh is needed so graphics can appear back on your computer.

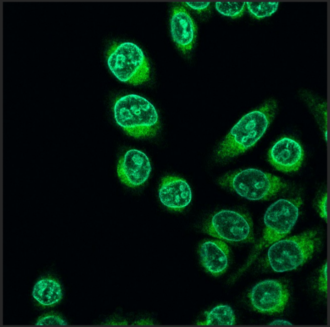
Testing in python

```
>>> import cv2
>>> print(cv2.__version__)
```

Basic Image Operations

Loading and displaying images using OpenCV.

```
>>> img = cv2.imread('exp_A01_G001_0008.oir.png')
>>> cv2.imshow("cargo", img) # may fail on Teach
>>> cv2.waitKey(0)           # so it shows up
>>> cv2.destroyAllWindows()
```



Source:
"Molecular determinants of large cargo transport into the nucleus"
https://idr.openmicroscopy.org/webclient/img_detail/9844391
DOI: 10.7554/eLife.55963

In Python, OpenCV images are Numpy arrays !

```
>>> print(type(img))
<class 'numpy.ndarray'>
>>> print(img.shape)
(583, 583, 3)
```

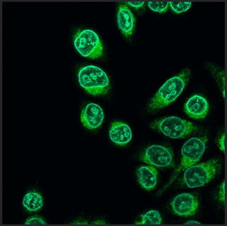
The latter are height, width and number of color channels.

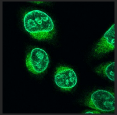
Accessing and modifying pixel values.

We can use all numpy's slicing and addressing tricks.

E.g. to crop to a portion of the image, slice it:

```
>>> h,w,c = img.shape
>>> cv2.imshow("image",img)
>>> subimg = img[h//4:-h//4,w//4:-w//4]
>>> cv2.imshow("sub-image",subimg)
>>> cv2.waitKey(0)
```

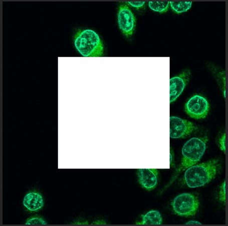




To set pixels, we can use `img[row,col]`. We can use slices as well.

E.g. to set our slice to all white:

```
>>> subimg[:,:] = (255,255,255)
>>> cv2.imshow("blanked-image", img)
>>> cv2.waitKey(0)
```



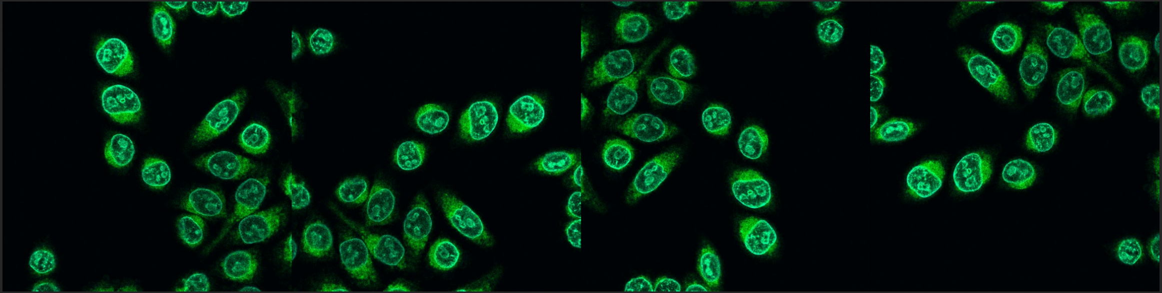
You can save the image with `cv2.imwrite("FILENAME",img)`.

Resizing and rotating images

To rotate by 90, 180, or 270 degrees:

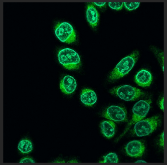
```
>>> img1 = cv2.imread("exp_A01_G001_0008.oir.png")
>>> img2 = cv2.rotate(img,cv2.ROTATE_90_CLOCKWISE)
>>> img3 = cv2.rotate(img,cv2.ROTATE_180)
>>> img4 = cv2.rotate(img,cv2.ROTATE_90_COUNTERCLOCKWISE)
```

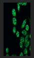
```
>>> import numpy as np
>>> cv2.imshow("rotate",np.hstack((img1,img2,img3,img4)))
>>> cv2.waitKey(0)
```



Resize an image

```
>>> print(img1.shape)
(583, 583, 3)
>>> img6 = cv2.resize(img,(0,0),fx=0.5,fy=0.5)
>>> print(img3.shape)
(292, 292, 3)
>>> img7 = cv2.resize(img,(40,80))
>>> print(img2.shape)
(80, 40, 3)
>>> cv2.imshow("small",img6)
>>> cv2.imshow("smaller",img7)
```





Understanding connection with Numpy

OpenCV works well in conjunction with numpy, but with an important subtlety:

- In computer vision images:
  - x coordinates run horizontal from left to right
  - y coordinates run vertical from top to bottom
  - x coordinates are given before y coordinates.
- In numpy matrices:
  - storage is row by row
  - the row index runs vertically
  - the column index runs horizontally
  - the rows are indexed first, then the columns second.

As a result, an openCV image size of `WIDTH` by `HEIGHT`
corresponds to a numpy array of shape `(HEIGHT,WIDTH)`.

I.e., coordinates given to opencv need to give the horizontal dimension first, vertical dimension second.
But working with the arrays, the vertical

Color Spaces

OpenCV supports a number of ways to encode colors:

- BRG (blue, green, red)
  The default storage in openCV.
- RGB (red, green, blue)
  More commonly used outsize of openCV, e.g., by matplotlib.
- HSV (hue, saturation, value)
  This is are better for picking out similar colors.
- Gray scale
  Often better for detecting shapes.

(In fact OpenCV has many more, these are the most common ones).

Converting between different color spaces

Color spaces and Matplotlib

Matplotlib's imshow can also plot images, but the colors would be off, because of the BGR ordering.

```
import matplotlib.pyplot as plt

plt.imshow(img) # wrong colours
plt.pause(.1)

plt.imshow(img[:,:,[2,1,0]]) # correct colours
plt.pause(.1)
```

Color conversions routines

```
>>> imgrgb = cv2.cvtColor(imgbgr, cv2.COLOR_BGR2RGB)
>>> imghsv = cv2.cvtColor(imgbgr, cv2.COLOR_BGR2HSV)
>>> imggry = cv2.cvtColor(imgbgr, cv2.COLOR_BGR2GRAY)
```

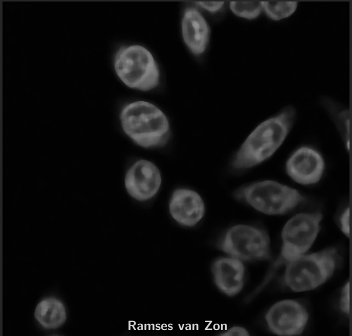
Image Filtering

Before processing an image, you may need to filter it or improve it.

Blur

```
>>> img = cv2.imread('exp_A01_G001_0008.oir.png), cv2.IMREAD_GRAYSCALE)
>>> imgmblur = cv2.medianBlur(img, 11)
>>> imggblur = cv2.GaussianBlur(img, (11,11),sigmaX=3.,sigmaY=3.)
```

The latter is weighted, the former better for edge detection.



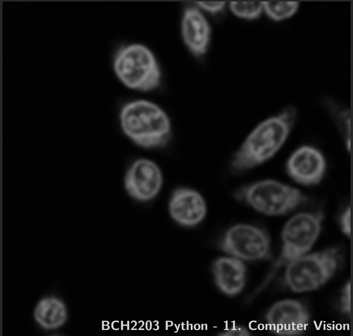
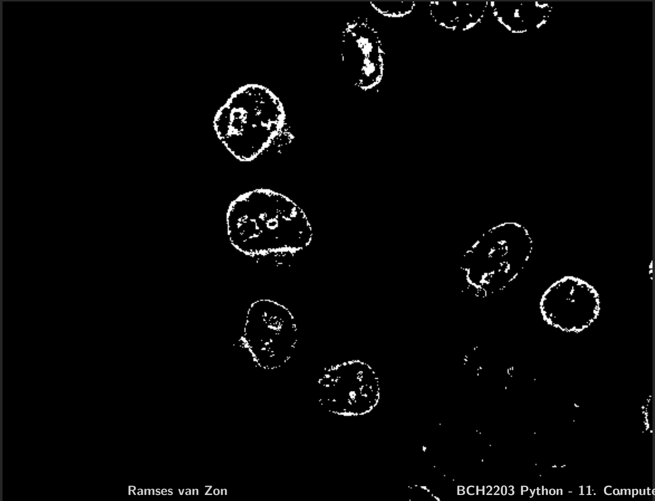


Image Thresholding

```
>>> img = cv2.imread('exp_A01_G001_0008.oir.png), cv2.IMREAD_GRAYSCALE)
>>> _, thresholded_image = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
```



Other features:

- Contours and Shape Detection
- Image Transformation and Perspective Correction
- Feature Detection and Description
- Object Detection
- Using pre-trained deep learning networks for object detection

And fun ones like:

- Webcam processing
- Video processing
- Face detection
- Motion detectin

Further Resources

- OpenCV's documentation is very extensive with lots of examples:

  https://docs.opencv.org/4.9.0/

- If you want some real microscopy images to play with, see

  https://idr.openmicroscopy.org/