Curs 4 (OpenCV)

https://pypi.org/project/opencv-python/

Keep in mind that Puppy Pi is running **Python 3.7.3** and **OpenCV 4.5.4.60**. If you use other versions on your pc, you may need to make some changes to make the code run.

The basic OpenCV code that should open the webcam and show on screen the live feed:

```
import cv2

# we will do our image processing in this method, so it will be easier to integrate on an ROS
# node later
def processImage(img):
    return img
if __name__ == "__main__":
    cv2.namedWindow("Camera")
    vc = cv2.VideoCapture(0)

    rval = True
    if not vc.isOpened():
        rval = False

    try:
        while rval:
            rval, frame = vc.read()
            result = processImage(frame)
            cv2.imshow("Camera", result)
            cv2.waitKey(1)
        except KeyboardInterrupt:
            cv2.destroyWindow("Camera")
            vc.release()
```

1. **Drawing basic shapes**
    a. Lines (https://docs.opencv.org/4.10.0/d6/d6e/group__imgproc__draw.html#ga7078a9fae8c7e7d13d24dac2520ae4a2)
    b. Rectangle (https://docs.opencv.org/4.10.0/d6/d6e/group__imgproc__draw.html#ga07d2f74cadcf8e305e810ce8eed13bc9)
    c. Circle (https://docs.opencv.org/4.10.0/d6/d6e/group__imgproc__draw.html#gaf10604b069374903dbd0f0488cb43670)
    d. Polygon (https://docs.opencv.org/4.10.0/d6/d6e/group__imgproc__draw.html#ga1ea127ffbbb7e0bfc4fd6fd2eb64263c)

e. Adding text (https://docs.opencv.org/4.10.0/d6/d6e/group__imgproc__draw.html#ga5126f47f883d730f633d74f07456c576)

2. **Image representation and properties:**
    a. Shape, size, data type

The shape of an image is a tuple of values indicating the height, width, and depth of an image.
The size of an image is simply the number of bytes in the image (height * width * depth)

    b. Split and merge color channels

We can split the image into its basic B, G, and R components, which are still treated as regular images, but with a depth of 1 (similar to grayscale). This may be useful when we want to process only one of the channels. Similarly, we can merge 3 2D matrices of the same shape and create a color image.
https://docs.opencv.org/4.10.0/d2/de8/group__core__array.html#ga0547c7fed86152d7e9d0096029c8518a
https://docs.opencv.org/4.10.0/d2/de8/group__core__array.html#ga7d7b4d6c6ee504b30a20b1680029c7b4

3. **Colorspaces**

https://docs.opencv.org/4.10.0/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab

    a. RGB/BGR

The most common colorspace is RGB, but OpenCV represents images internally as BGR. In a regular scenario, opening an image or capturing a frame using OpenCV automatically represents it in the BGR colorspace. However, if we have an image as an array of bytes (as is the case for the images received on the **/usb_cam/image_raw** topic), we must ensure that the BGR representation is obeyed.

    b. Other colorspaces (Lab, YCrCb, HSV, Grayscale)

- Lab (Lightness, color component ranging from Green to Magenta, color component ranging from Blue to Yellow)
    - The Lab color space is quite different from the RGB color space. In RGB color space the color information is separated into three channels but the same three channels also encode brightness information. On the other hand, in the Lab color space, the L channel is independent of color information and encodes brightness only. The other two channels encode color.
- YCrCb (Luminance, red chroma, blue chroma
    - Separates the luminance and chrominance components into different channels.
    - Mostly used in compression (of Cr and Cb components) for TV Transmission.
    - Device dependent.
- HSV (Hue - dominant wavelength, Saturation - shades of color, Value - intensity of the color)
    - The most notable thing is that it uses only one channel to describe color (H), making it very intuitive to specify color.
- Grayscale
    - Contains only one channel, thus it reduces the image size
    - Useful when color is not needed

4. **Geometric transformations**
    a. Resizes

(https://docs.opencv.org/4.10.0/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d)

    b. Affine transformations

(https://docs.opencv.org/4.10.0/da/d54/group__imgproc__transform.html#ga0203d9ee5fcd28d40dbc4a1ea4451983)

Affine transformations will warp the image with the restriction that all parallel lines from the original image will remain parallel in the result. To perform an affine transformation we need the coordinates of 3 points in the original image and the coordinates of the new location of each of the three points in the result:

pts1 = np.float32([[50,50],[200,50],[50,200]])

pts2 = np.float32([[10,100],[200,50],[100,250]])

We use getAffineTransform to let OpenCV calculate the affine transformation matrix (a 2 x 3 matrix) that is required by warpAffine:

M = cv.getAffineTransform(pts1,pts2)

We apply the warp to the image.

dst = cv.warpAffine(img,M,(cols,rows))

The affine transformation can be represented by the following function, where x and y are the coordinates of a point in the original image and M_{11} - M_{23} are the elements of matrix M.

$dst(x,y) = src(M_{11}x+M_{12}y+M_{13},M_{21}x+M_{22}y+M_{23})$

5. **Noise and filtering**

- Noise refers to unwanted or random fluctuations in the pixel values of an image. Noise can be caused by various factors:
    - hardware: issues with the imaging sensor, electrical interference, transmission errors, etc.
    - environmental: lighting conditions, weather, etc.

Voluntarily adding noise to images may seem like a counterintuitive operation, however, for training an artificial intelligence model, expanding a dataset of images with images with noise may increase its performance with real data.

- Some of the more common ways of adding noise to an image are:
    - Gaussian noise
    - Salt and pepper noise
    - Random noise
- Filtering is the operation that smoothes an image and attempts to reduce the amount of noise by removing high-frequency content. Some of the common noise removal operations are:
    - Average blurring (https://docs.opencv.org/4.10.0/d4/d86/group__imgproc__filter.html#ga8c45db9afe636703801b0b2e440fce37)

- Gaussian blurring (https://docs.opencv.org/4.10.0/d4/d86/group__imgproc__filter.html#gaabe8c836e97159a9193fb0b11ac52cf1)
- Median blurring (https://docs.opencv.org/4.10.0/d4/d86/group__imgproc__filter.html#ga564869aa33e58769b4469101aac458f9)
- Bilateral filter -> attempts to preserve edges (https://docs.opencv.org/4.10.0/d4/d86/group__imgproc__filter.html#ga9d7064d478c95d60003cf839430737ed)

6. Morphological operations

A set of operations that process images based on shapes. Morphological operations apply a *structuring element* to an input image and generate an output image. The most basic morphological operations are Erosion and Dilation. They have a wide array of uses:

- Removing noise
- Isolation of individual elements and joining disparate elements in an image.
- Finding intensity bumps or holes in an image

a. Erosion and dilation

- Dilation:
- https://docs.opencv.org/4.10.0/d4/d86/group__imgproc__filter.html#ga4ff0f3318642c4f469d0e11f242f3b6c
    - This operation consists of convolving an image with some kernel (K), which can have any shape or size, usually a square or circle. The kernel has a defined *anchor point*, usually being the center of the kernel. As the kernel is scanned over the image, we compute the maximal pixel value overlapped by and replace the image pixel in the anchor point position with that maximal value. As you can deduce, this maximizing operation causes bright regions within an image to "grow" (therefore the name *dilation*).
- Erosion:
- https://docs.opencv.org/4.10.0/d4/d86/group__imgproc__filter.html#gaeb1e0c1033e3f6b891a25d0511362aeb
    - This operation is complementary to dilation. It computes a local minimum over the area of a given kernel. As the kernel is scanned over the image, we compute the minimal pixel value overlapped by and replace the image pixel under the anchor point with that minimal value.

b. Opening and closing

https://docs.opencv.org/4.10.0/d4/d86/group__imgproc__filter.html#ga674937763ad1a3df63883829375201f

- The opening operation is an erosion followed by a dilation. This operation can remove or diminish bright spots in an image.
- The closing operation is a dilation followed by an erosion. It is useful in closing small holes inside the foreground objects, or small dark spots on an object.

7. **Thresholding**

Thresholding is the process of separating the pixels that have a value within a specified range from those that are outside of that range.

a. Binarization

https://docs.opencv.org/4.10.0/d7/d1b/group__imgproc__misc.html#gae8a4a146d1ca78c626a53577199e9c57

In the case of binarization, pixels with a value belonging to a specified [X, Y] interval become white and those outside that range become black. (Inverse binarization is also available which turns any pixel within the [X, Y] range black and those outside the range white). It is recommended to apply binarization on grayscale images instead of color ones, as in the case of color images, the thresholding is applied on each channel separately.

b. In range masking

https://docs.opencv.org/4.10.0/d2/de8/group__core__array.html#ga48af0ab51e36436c5d04340e036ce981

In range masking can isolate color pixels with N channels by ensuring that all components of a pixel are bounded by an N-dimensional vector of values. It is recommended to apply the masking to a colorspace that contains the color information of fewer channels (eg. HSV or Lab).

8. **Contour detection**

https://docs.opencv.org/4.10.0/d4/d73/tutorial_py_contours_begin.html

https://docs.opencv.org/4.10.0/d3/dc0/group__imgproc__shape.html#gae4156f04053c44f886e387cff0ef6e08