International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2016 IJSRCSEIT | Volume 1 | Issue 2 | ISSN : 2456-3307

An Introduction to OpenCV using Python with Ubuntu

Krupali Mistry[1], Avneet Saluja[2]

[1,2]Assistant Professor,Department of Computer Science and Engineering, ITM Universe, Vadodara, Gujarat, India

ABSTRACT

OpenCV is an open source library for image and video analysis by Intel.It is related to computer vision, such as feature and object detection and machine learning.The purpose of the paper is implementation of image processing techniques using opencv with python in Ubuntu platforms.

Keywords: Opencv, Ubuntu, Python, Image Processing Techniques.

I. INTRODUCTION

Computer vision is a fast growing field for analyzing, modifying, and high-level understanding of images. Main motive is to determine and understand the images and to provide people with new images that are extra useful.

OpenCV is an Image Processing library created by Intel,It can be freely downloaded and It is available for C, C++, Java and Python,Windows, Linux, Mac OS, iOS and Android Newest Its versions are opencv-2.4.13 and opencv-3.1.It is open Source and also Easy to use and install .It is designed for computational efficiency strong focus on real-time applications The first implementation was in the **C** programming language; however, its popularity grew with its C++ implementation as of Version 2.0. New functions are programmed with C++. However, nowadays, the library has a full interface for other programming languages, such as Java, Python, and MATLAB/Octave.OpenCV is freely available for download at http://opencv.org. This site provides the last version for distribution (currently, 3.0 beta) and older versions. OpenCV requires that images be in BGR or Grayscale in order to be shown or saved. Otherwise, undesirable effects may result.

II. METHODS AND MATERIAL

Open cv Features:

- Both low and high level API
- Image data manipulation (allocation, release, copying, setting, conversion).
- Image and video I/O (file and camera basedinput,image/video file output).
- Matrix and vector manipulation and linear algebra routines.
- Various dynamic data structures (lists, queues, sets, trees, graphs).
- Basic image processing (filtering, edge detection, corner detection, samplingand interpolation, color conversion, morphological operations, histograms,image pyramids
- Structural analysis (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation).
- Camera calibration (finding and tracking calibration patterns, calibration, fundamental matrix estimation, homography estimation, stereo correspondence).
- Motion analysis (optical flow, motion segmentation, tracking).
- Object recognition (eigen-methods, HMM).
- Basic GUI (display image/video, keyboard and mouse handling, scroll-bars).Image labeling (line, conic, polygon, text drawing).

Modules of opencv

- **cv -** Main OpenCV functions

- **cvaux** - Auxiliary (experimental) OpenCV functions
- **cxcore -** Data structures and linear algebra support
- **highgui** - GUI functions, This provides simple **user interface** (**UI**) capabilities.
- **imgproc** - linear and non-linear image filtering, geometrical image transformations,
  color space conversion, histograms, etc.
- **calib3d** - multiple-view geometry algorithms, single and stereo camera calibration, etc.
- **features2d** - These are functions for feature detection (corners and planar
- objects), feature description, feature matching, and so on.
- **objdetect** - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, etc.)
- **gpu** - GPU-accelerated algorithms from different OpenCV modules
- **others** – helper

Basic opencv Data Structures

- **Point, Point2f -** 2D Point: *int x,y* several operators and fuctions available
- **Size -** 2D size structure: *int width, height*
- **Rect -** 2D rectangle object *int x, y, width, height*
- **RotatedRect -** Rect object with angle
- **Mat** - image object mainly - *rows, cols - length and width, channels, depth*,a large number of functions for manipulation, a critical being Mat.type() returns the TYPE of a matrix

OpenCV: Types

The TYPE is a very important aspect of OpenCV Represented as CV_<Datatype>C<no of Channels>
OpenCV: PixelTypes
shows how the image is represented in data

- BGR - The default color of imread(). Normal 3 channel color
- HSV - Hue is color, Saturation is amount, Value is lightness. 3 channels
- GRAYSCALE - Gray values, Single channel
- Converting colorspaces: cvtColor(image, image, code): codes

CV_<colorspace>2<colorspace>,

e.g CV_BGR2GRAY, CV_BGR2HSV

Basic Image Transformation Functions or Techniques of opencv

All OpenCV classes and functions are in the cv namespace, and consequently, we will have the following two options in our source code:

- Add the using namespace cv declaration after including the header files.
- Append the cv:: prefix to all the OpenCV classes, functions, and data structures that we use. This option is recommended if the external names provided by OpenCV conflict with the often-used **standard template library** (**STL**) or other libraries.

Image Normalization: process of stretching the range of an image from [a, b] to [c, d], important for visualization, normalize(imagein, imageout, low, high, method). This is incredibly important for visualization because if the image is beyond [0,255] it will cause truncation or unsightly effects.

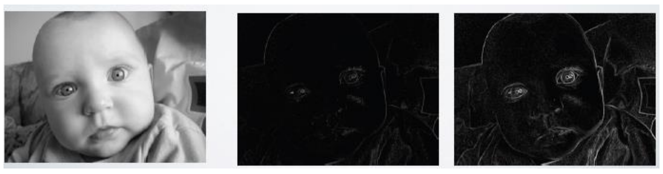
Figure 1. Image Normalization Results

Thresholding threshold(image, image, thresh, maxVal, CODE), codes e.g. THRESH_BINARY

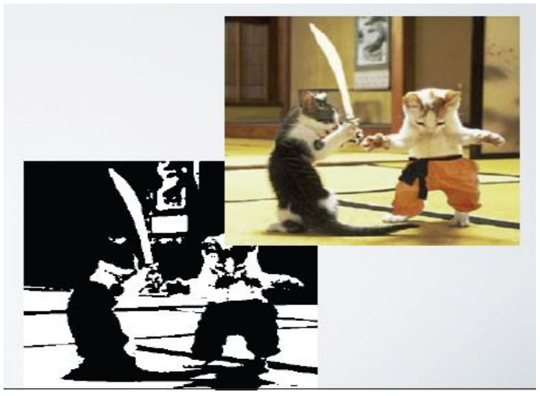
Figure 2. Image Thresholding Results

Edge Detection: several methods available: Sobel, Scharr, Laplacian and Canny
• Sobel Edge Detection
void cv::Sobel(image in, image out, CV_DEPTH, dx, dy);
• Scharr Edge Detection

void cv::Scharr(image in, image out, CV_DEPTH, dx, dy);
• Laplacian Edge Detection
void cv::Laplacian(image in, image out, CV_DEPTH);

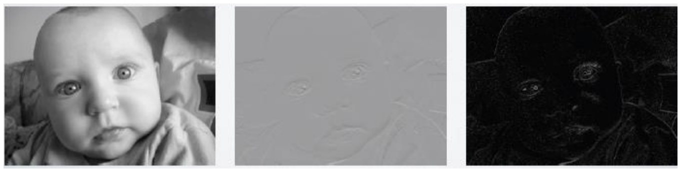
Figure 3. Edge Detection Results

Image Smoothing: Image smoothing is used to reduce the the sharpness of edges and detail in an image. OpenCV includes most of the commonly used methods.
• void **GaussianBlur**(imagein, imageout, Size ksize, *sig);*
• *void* **medianBlur** (imagein, imageout, Size ksize);

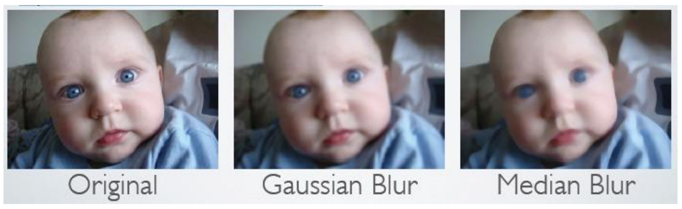

Figure 4. Image Smoothing Results

Erosion:
Erosion erodes the image. It tries to bring uniformity in the image by converting bright points in neighborhood of points of less intensity into darker ones

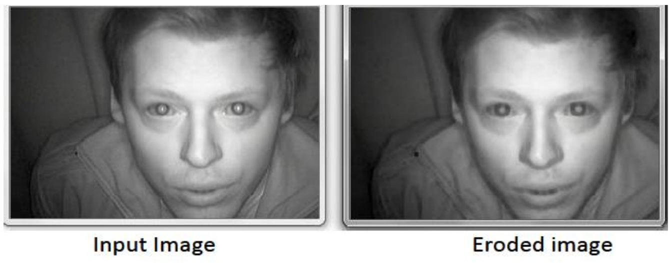

Figure 5. Image Erosion Results

Notice the change in eyes, illuminates spots in the eyes are removed because in the input image there is a stark change in illumination at points near pupil.

Dilation:

Dilation dilates the image. It tries to bring uniformity in image by converting dark points in neighborhood of points of higher intensity into bright ones.

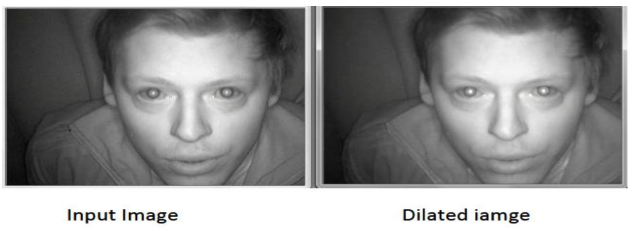

Figure 6. Image Dilation Results

III. RESULTS AND DISCUSSION

All
fully in Regular font.

IV. CONCLUSION

This paper presented a brief look at implementing basic computer vision functionality using Python. The three libraries looked at have very different goals, but can all be used for computer vision at different levels. OpenCV can be used for computer vision development and applications.For embedded platforms where speed is of utmost importance, or when computer vision functionality is the main requirement, OpenCV is the fastest and most complete tool for computer vision. OpenCV is considered by many to be side by side with many commercial image-processing packages, and yet it is an open source tool. Furthermore thanks to the fact that OpenCV keeps evolving is an additional guarantee that it will advance research in vision and promote the development of rich, vision-based CPU intensive applications.

V. REFERENCES

[1] http://www.sourceforge.net/projects/opencvlibrary Computer VisionHomepage
[2] http://www-2.cs.cmu.edu/~cil/vision.html
[3] R. Szeliski. Computer Vision: Algorithms and Applications. Springer 2011.
[4] http://opencv.willowgarage.com/wiki/
[5] http://en.wikipedia.org/wiki/BSD_license
[6] R. Laganière. OpenCV 2 Computer Vision Application Programming Cookbook. Packt Publishing 2011
[7] J. Canny. A Computational approach to edge detection. IEEE Transactions on Pattern Analysis and Image Understanding, vol.18 (6), pp. 679-698, 1986.

[8] C. Harris, M.J. Stephens. A combined corner and edge detector. Alvey Vision Conference, pp. 147–152, 1988.
[9] J. Shi and C. Tomasi. Good Features to Track. IEEE Conference on Computer Vision and Pattern Recognition, pp. 593-600, 1994.
[10] Learning OpenCV" by Gary Bradski and Adrian Kaehler;
[11] "OpenCV 2 Computer Vision Application Programming Cookbook" by Robert Laganiere;
[12] Digital Image Processing" by Rafael C. Gonzalez and Richard E. Woods;
[13] Digital Image Processing By Gonzalez2ndEdition2002.pdf