**OPENCV NOTES**

**CORE FUNCTIONALITIES**

* **Basic Structures**
  + **cv::InputArray**
    - * A proxy class for passing read only input arrays to opencv functions
      * is a class that can be constructed from [**Mat**](https://docs.opencv.org/4.x/d3/d63/classcv_1_1Mat.html), [**Mat\_**](https://docs.opencv.org/4.x/df/dfc/classcv_1_1Mat__.html)<T>, **[Matx](https://docs.opencv.org/4.x/de/de1/classcv_1_1Matx.html" \o "Template class for small matrices whose type and size are known at compilation time. )**<T, m, n>, std::vector<T>, std::vector<std::vector<T> >, std::vector<[**Mat**](https://docs.opencv.org/4.x/d3/d63/classcv_1_1Mat.html)>, std::vector<[**Mat\_**](https://docs.opencv.org/4.x/df/dfc/classcv_1_1Mat__.html)<T> >, **[UMat](https://docs.opencv.org/4.x/d7/d45/classcv_1_1UMat.html)**, std::vector<**[UMat](https://docs.opencv.org/4.x/d7/d45/classcv_1_1UMat.html)**> or double. It can also be constructed from a matrix expression.
        + When you see in the reference manual or in OpenCV source code a function that takes InputArray, it means that you can actually pass Mat, Matx, vector<T> etc. (see above the complete list).
        + Optional input arguments: If some of the input arrays may be empty, pass cv::noArray() (or simply cv::Mat() as you probably did before).
        + The class is designed solely for passing parameters. That is, normally you should not declare class members, local and global variables of this type.
        + If you want to design your own function or a class method that can

operate of arrays of multiple types, you can use **InputArray** (or **OutputArray**) for the respective parameters. Inside a function you should use **\_InputArray::getMat()** method to construct a matrix header for the array (without copying data). **\_InputArray::kind()** can be used to distinguish Mat from vector<> etc., but normally it is not needed.

* + - * There is another related type, **InputArrayOfArrays**, which is currently defined as a synonym for InputArray: typedef [InputArray](https://docs.opencv.org/4.x/dc/d84/group__core__basic.html#ga353a9de602fe76c709e12074a6f362ba) [InputArrayOfArrays](https://docs.opencv.org/4.x/dc/d84/group__core__basic.html#ga606feabe3b50ab6838f1ba89727aa07a);
        + It denotes function arguments that are either vectors of vectors or vectors of matrices. At the function implementation level their use is similar, but **\_InputArray::getMat(idx)** should be used to get header for the idx-th component of the outer vector and **\_InputArray::size().area()** should be used to find the number of components (vectors/matrices) of the outer vector.
  + **cv::\_OutputArray**
    - * This type is very similar to InputArray except that it is used for input/output and output function parameters.
  + **Cv::Algorithm**
    - * This is a base class for all more or less complex algorithms in OpenCV.
      * This class also consist of derived class that could be useful for my project, Threat Detection
  + **cv::Complex< \_Tp >**
    - * A complex number class
      * Similar to **std::complex**, however it provides slightly more convenient access to the real and imaginary parts using through the simple field access, as opposite to **std::complex::real()**  and **std::complex::imag().**
      * **Complex Numbers detailed explanation link:** [Complex Numbers • Computer Science and Machine Learning (xarg.org)](https://www.xarg.org/book/analysis/complex-numbers/)
  + **cv::DataType<\_Tp>** 
    - * The main purpose of this class is to convert compilation-time type information to an OpenCV-compatible data type identifier.
      * [OpenCV: cv::DataType< \_Tp > Class Template Reference](https://docs.opencv.org/4.x/d0/d3a/classcv_1_1DataType.html)
      * This class is a “trait” class for OpenCV primitive data types
      * Such traits are used to tell the OpenCV which data types you are working with, even if such data type is not native to OpenCV
      * This also useful for generic algorithms implementations.
  + **cv::DMatch**
    - * A class for matching keypoint descriptors.
      * **Keypoints and Descriptors in depth explanation link:** [16. Keypoints and Descriptors - Learning OpenCV 3 [Book] (oreilly.com)](https://www.oreilly.com/library/view/learning-opencv-3/9781491937983/ch16.html#:~:text=The%20descriptive%20information%20about%20a%20keypoint%20is%20summarized,OpenCV%203%20now%20with%20the%20O%E2%80%99Reilly%20learning%20platform.)
        + The tracking of corners evolved into theory of keypoints. The concept of corners and keypoints, is based on intuition that it would be useful in many applications to be able to represent an image or object in an invariant form that will be the same, or at least very similar, in other similar images of the same scene or object. Corner and keypoint representations are powerful methods for doing this.
        + A corner is a small patch of image that is rich in local information and therefore likely to be recognized in another image.
        + A **keypoint** is an extension of this concept that encodes information from a small local patch of an Image such that keypoint is highly recognizable and, at least in principle, largely unique.
        + The descriptive information about a keypoint is summarized in the form of its descriptors, which is typically much lower-dimensional than the pixel patch that formed the keypoint. The descriptor represents that patch so as to make it easier.
        + One important thing to understand is that after extracting the keypoints, you only obtain information about their position, and sometimes their coverage area (usually approximated by a circle or ellipse) in the image. While the information about keypoint position might sometimes be useful, it does not say much about the keypoints themselves. Source: [Meaning of Keypoints and Descriptors - OpenCV Q&A Forum](https://answers.opencv.org/question/37985/meaning-of-keypoints-and-descriptors/)

[OpenCV: cv::Mat Class Reference](https://docs.opencv.org/4.x/d3/d63/classcv_1_1Mat.html)