|  |  |
| --- | --- |
| **Function** | **Purpose** |
| imread(image\_path, flag) | This method is used to read an image from its path |
| imshow(window\_name, image) | It is used to show the image in the window. |
| imwrite(filename, image) | This method is used to write or save an image using OpenCV. |
| **Image Reading Modes** | **Description** |
| cv2.IMREAD\_COLOR | Read colored images |
| cv2.IMREAD\_GRAYSCALE | Read Grayscale images |
| cv2.IMREAD\_UNCHANGED | Read images with alpha channel |

### **Showing Images**

The [cv2.imshow()](https://www.geeksforgeeks.org/python-opencv-cv2-imshow-method/) method is used to display an image in a window.

cv2.imshow(window\_name, image)

### **Image Cropping**

Cropping is the removal of unwanted outer areas from an image.

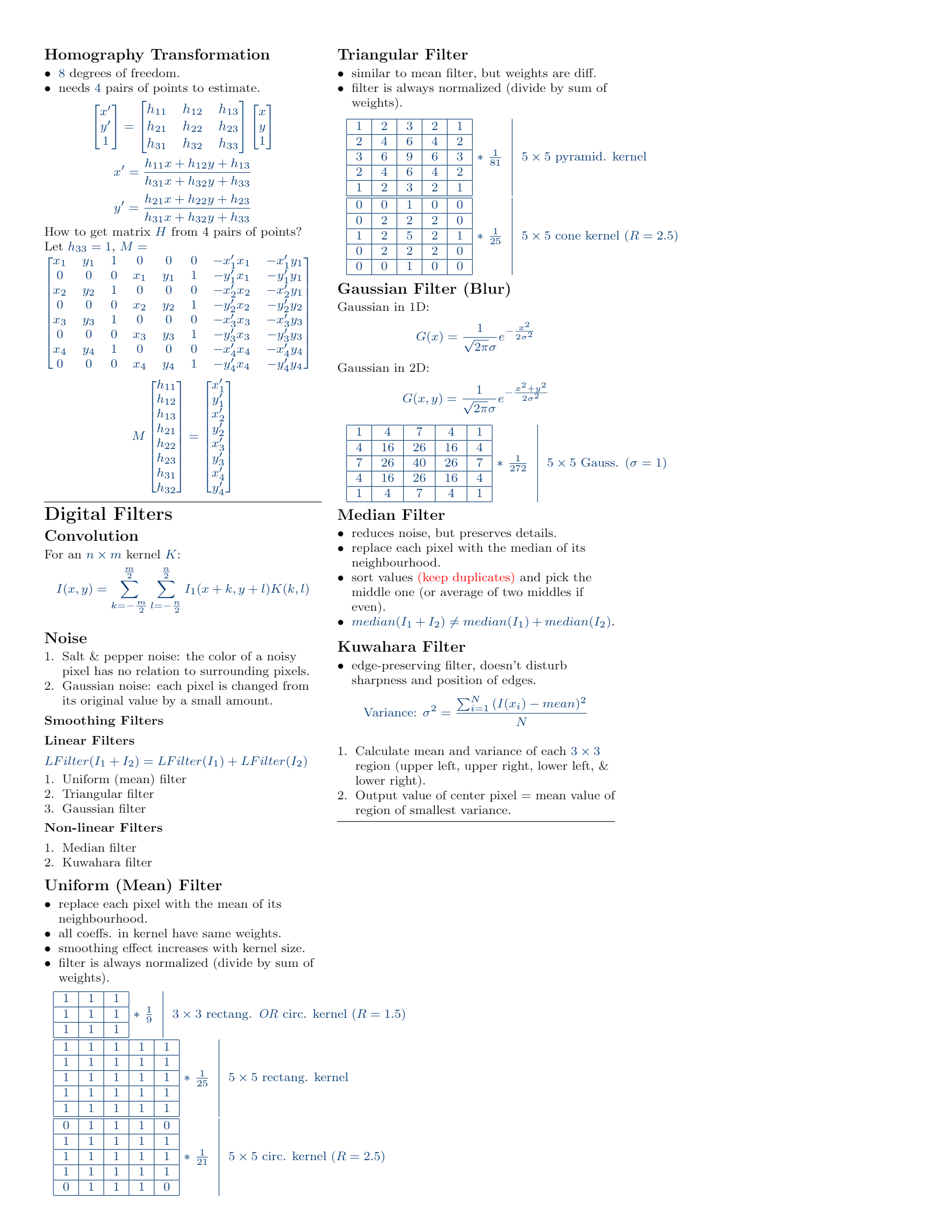
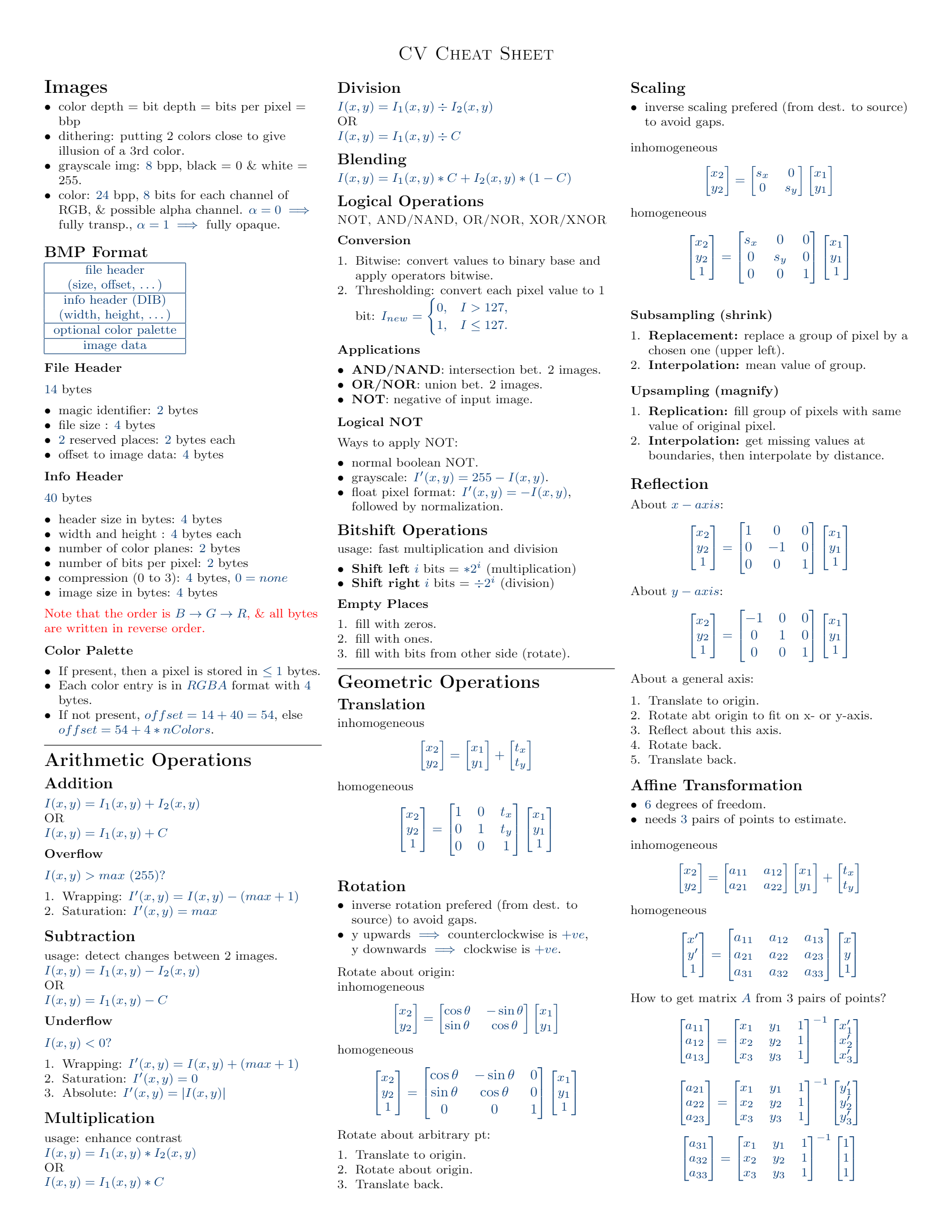
cropped\_img = img[100:300, 100:300]

### **Saving an Image**

The [cv2.imwrite()](https://www.geeksforgeeks.org/python-opencv-cv2-imwrite-method/) method is used to save an image to any storage device. This will save the image according to the specified format in the current working directory.

cv2.imwrite(filename, image)

|  |  |
| --- | --- |
| **Function** | **Uses** |
| add(image1, image2) | This function is used to add two images. |
| subtract(image1, image2) | This function is used to subtract two images. |
| addWeighted(image1, weight1, image2, weight2, gammaValue) | This is also known as Alpha Blending. This is nothing but a weighted blending process of two images. |
| bitwise\_and(image1, image2, destination, mask) | This performs bitwise and logical operations between two images. |
| bitwise\_or(image1, image2, destination, mask) | This performs bitwise or logical operations between two images. |
| bitwise\_not(image, destination, mask) | This performs bitwise not logical operations between an image and a mask. |
| bitwise\_xor(image1, image2, destination, mask) | This performs bitwise xor logical operations between two images. |
| **Scaling Types** | **Description** |
| cv2.INTER\_AREA | Used to shrink the image |
| cv2.IN­TER­\_CUBIC | Bicubic interpolation |
| cv2.INTER\_LINEAR | Default interpolation technique used to zoom the image |
| **Operation** | **Uses** |
| cv2.Canny(image, T\_lower, T\_upper, aperture\_size, L2Gradient) | The [Canny Edge Detection](https://www.geeksforgeeks.org/implement-canny-edge-detector-in-python-using-opencv/) is an algorithm used for edge detection. It reduces noise using Gaussian Smoothing and computes image gradient using The Sobel filter. |
| cv2.HoughLines(edges, 1, np.pi/180, 200) | The Hough Transform is a method that is used in image processing to detect any shape if that shape can be represented in mathematical form. It can detect the shape even if it is broken or distorted a little bit. |
| cv2.HoughCircles(image, cv2.HOUGH\_GRADIENT, 1, 20, param1 = 50, param2 = 30, minRadius = 1, maxRadius = 40) | [Circle detection](https://www.geeksforgeeks.org/circle-detection-using-opencv-python/) finds a variety of uses in biomedical applications, ranging from iris detection to white blood cell segmentation. |
| cv2.cornerHarris(src, dest, blockSize, kSize, freeParameter, borderType) | [Harris Corner detection](https://www.geeksforgeeks.org/python-corner-detection-with-harris-corner-detection-method-using-opencv/) algorithm was developed to identify the internal corners of an image. The corners of an image are basically identified as the regions in which there are variations in the large intensity of the gradient in all possible dimensions and directions. |
| cv2.goodFeaturesToTrack(image, max\_corner, quantity\_level, min\_distance) | The cv2.goodFeaturesToTrack() function in OpenCV is used for corner detection. |
| cv2.drawKeypoints(input\_image, key\_points, output\_image, colour, flag) | The distinguishing qualities of an image that make it stand out are referred to as key points in an image. The key points of a particular image let us recognize objects and compare images. This can be done by using the cv2.drawKeypoints() method. |



**Installation Instructions:**

* **pip install tensorflow**
* **pip install keras**
* **pip install opencv-python**

**Basic Usage Examples:**

* TensorFlow: **import tensorflow as tf**
* Keras: **import keras**
* OpenCV: **import cv2**

**Ambalika**:

**Cheat Sheet: Object Detection Basics**

**Key Concepts:**

**Bounding Boxes**: Rectangular regions defining objects.

• **Definition**: Bounding boxes are represented as (x, y, width, height) coordinates specifying the location and size of detected objects within an image.

• **Formula**: Bounding box coordinates: (x, y, w, h)

**Annotations**: Labeling objects with bounding boxes and class labels.

• **Definition**: Annotations involve manually or automatically marking objects in images with bounding boxes and corresponding class labels.

**Confidence Scores:** Probability scores indicating prediction confidence.

• **Definition**: Confidence scores represent the model's certainty in detecting an object.

• **Formula**: Confidence score (c) ∈ [0, 1]

**IoU (Intersection over Union):** Measure of overlap between two bounding boxes.

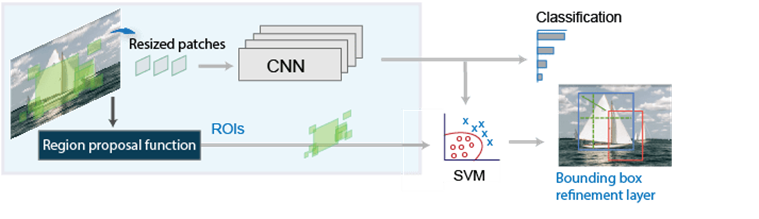
• **Definition**: IoU calculates the ratio of the area of intersection to the area of union between two bounding boxes.

• **Formula**: IoU = Area of Intersection / Area of Union

**Object Detection Algorithms:**

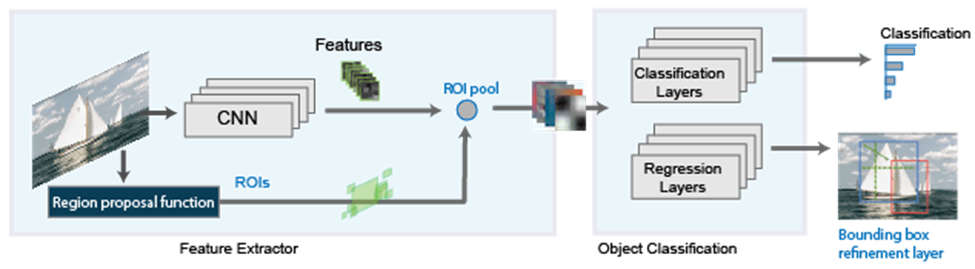
• **R-CNN** The R-CNN detector first generates region proposals using an algorithm such as Edge Boxes. The proposal regions are cropped out of the image and resized. Then, CNN classifies the cropped and resized regions. Finally, the region proposal bounding boxes are refined by a support vector machine (SVM) that is trained using CNN features.

Use the [trainRCNNObjectDetector](https://www.mathworks.com/help/vision/ref/trainrcnnobjectdetector.html) function to train an R-CNN object detector. The function returns an [rcnnObjectDetector](https://www.mathworks.com/help/vision/ref/rcnnobjectdetector.html) object that detects objects in an image.

**Diagram**:

• **Fast R-CNN** As in the R-CNN detector, the Fast R-CNN detector also uses an algorithm like Edge Boxes to generate region proposals. Unlike the R-CNN detector, which crops and resizes region proposals, the Fast R-CNN detector processes the entire image. Whereas an R-CNN detector must classify each region, Fast R-CNN pools CNN features corresponding to each region proposal. Fast R-CNN is more efficient than R-CNN because in the Fast R-CNN detector, the computations for overlapping regions are shared.

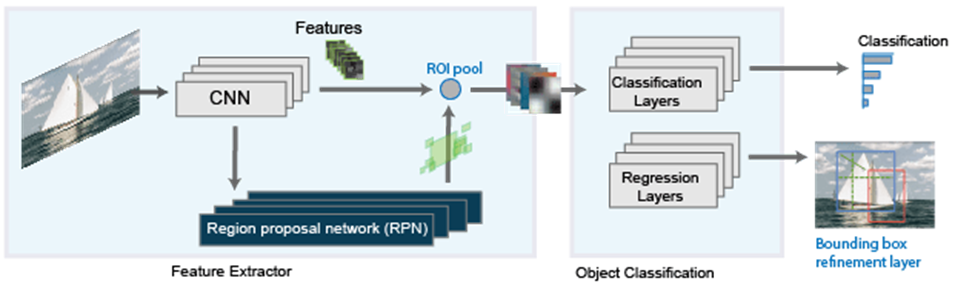
Use the [trainFastRCNNObjectDetector](https://www.mathworks.com/help/vision/ref/trainfastrcnnobjectdetector.html) function to train a Fast R-CNN object detector. The function returns a [fastRCNNObjectDetector](https://www.mathworks.com/help/vision/ref/fastrcnnobjectdetector.html) that detects objects from an image.

• **Diagram**: 

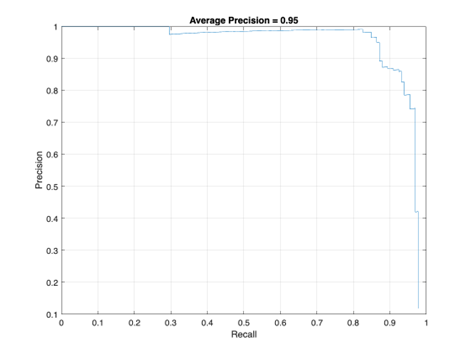
• **Faster R-CNN:** Incorporates Region Proposal Network (RPN).

The Faster R-CNN detector adds a region proposal network (RPN) to generate region proposals directly in the network instead of using an external algorithm like Edge Boxes. The RPN uses [Anchor Boxes for Object Detection](https://www.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html). Generating region proposals in the network is faster and better tuned to your data.

Use the [trainFasterRCNNObjectDetector](https://www.mathworks.com/help/vision/ref/trainfasterrcnnobjectdetector.html) function to train a Faster R-CNN object detector. The function returns a [fasterRCNNObjectDetector](https://www.mathworks.com/help/vision/ref/fasterrcnnobjectdetector.html) that detects objects from an image.

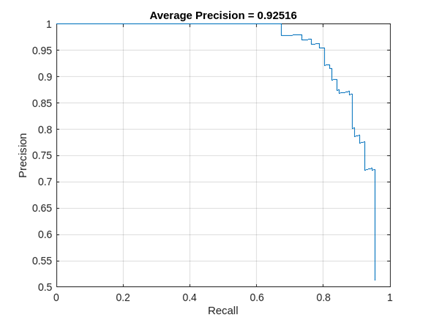
• **Diagram**: 

• **SSD**: The Single Shot MultiBox Detector (SSD) is a robust object detection framework based on a feed-forward convolutional neural network (CNN). Design SSD's architecture to generate a fixed set of bounding boxes and associated scores, indicating the presence of object class instances in those boxes. Use [trainSSDObjectDetector](https://www.mathworks.com/help/vision/ref/trainssdobjectdetector.html) function to train SSD object detector if doTraining to true. Otherwise, load a pretrained network.



• **YOLO**: Direct prediction of bounding boxes and class probabilities.

YOLO is a popular and fast object detection algorithm that uses a single neural network to predict bounding boxes and class probabilities for objects in an image. Train a YOLO v3 object detection network using the [trainYOLOv3ObjectDetector](https://www.mathworks.com/help/vision/ref/trainyolov3objectdetector.html) (Computer Vision Toolbox)function. This function mitigates the need to write a custom training loop and supports advanced operations such as parallel and multi-GPU training.



In conclusion, object detection is a computer vision task that uses deep learning techniques to detect objects in images and videos.

**Object Detection Task Steps:**

1. **Data Collection:** Obtain labeled images with annotations.

2. **Preprocessing**: Resize, normalize images.

3. **Model Selection:** Choose suitable detection model.

4. **Training**: Fine-tune model on labeled dataset.

5. **Evaluation**: Measure performance using metrics like IoU.

6. **Inference:** Apply trained model for object detection.

**Common Challenges and Troubleshooting:**

• **Overfitting**: Regularization, data augmentation.

• **Data Imbalance**: Class weighting, augmentation techniques.

• **Performance**: Hyperparameter optimization, model architecture.

• **Runtime Efficiency**: Model quantization, hardware acceleration.

* **Data Imbalance:** Address class imbalance by using techniques like data augmentation or weighted loss functions.
* **Overfitting:** Regularize the model with techniques like dropout or early stopping.
* **Poor Localization:** Experiment with different network architectures or adjust hyperparameters related to bounding box regression.

**Common Applications of Object Detection:**

* Self-driving cars | Cars navigate by identifying pedestrians, vehicles, and traffic signs. | Bounding boxes, confidence scores | Faster R-CNN (accuracy), SSD (real-time) | TensorFlow, OpenCV
* Facial Recognition | Facial recognition systems detect faces for identification or unlocking devices. | Bounding boxes. | R-CNN (high accuracy – facial features), Faster R-CNN (faster alternative) | TensorFlow, Keras, OpenCv
* Medical Scan (X-ray, CT scan) | Doctors leverage detection for abnormalities like tumors or fractures. | Bounding boxes, annotations. | faster R-CNN (accuracy for small objects), YOLO (speed for large scans) | TensorFlow, OpenCV
* Inventory | Inventory Management with automatic stock checks and reordering. | Bounding Boxes, object classification. | SSD, YOLO (efficiency) | TensorFlow, Keras
* Robotics | Accurate perception of surroundings for tasks like picking and placing objects. | Bounding boxes, depth perception. | Faster R-CNN (accuracy), SSD (real-time for some tasks) | TensorFlow, OpenCV

**Concepts:**

* Bounding Boxes: Rectangular boxes drawn around detected objects to indicate their location.
* Annotations: Labels assigned to bounding boxes, specifying the object type (“car”, “person”)
* Confidence Scores: values between 0 and 1 indicating the model’s certainty about the detection.
* Intersection over Union (IoU): A metric used to measure how well a predicted bounding box overlaps with the ground truth box (actual location).

**Common Algorithms:**

* R-CNN (Regions with CNN features): A foundational algorithm (slower) but good for high accuracy.
* Fast R-CNN: Improves speed over R-CNN.
* Faster R-CNN: Faster and more accurate than Fast R-CNN, good for tasks requiring high precision.
* SSD (Single Shot MultiBox Detector): Fast, one-shot detection, ideal for real-time applications.
* YOLO (You Only Look Once): Very fast, one-shot detection, excellent for real-time applications (may have lower accuracy on small objects).

**Tools and Libraries:**

* **TensorFlow:** A popular open-source framework for machine learning, widely used for object detection tasks. Deep learning framework with built-in support for object detection models like SSD and Faster R-CNN.
* **Keras:** A high-level API built on top of TensorFlow, simplifying the development process for rapid prototyping and experimentation.
* **OpenCV** (Open-Source Computer Vision Library): A library offering computer vision functions and real-time capabilities, often used for object detection preprocessing and postprocessing tasks and providing tools for image processing, feature detection, and object tracking.

**References:**

* TensorFlow: [[API Documentation | TensorFlow v2.16.1](https://www.tensorflow.org/api_docs)]
* Keras: [[Developer guides (keras.io)](https://keras.io/guides/)]
* OpenCV: [[OpenCV: OpenCV modules](https://docs.opencv.org/4.x/index.html)]