```
!pip install clodsa
```

```
Collecting clodsa
  Downloading clodsa-1.2.47.tar.gz (30 kB)
Collecting mahotas
  Downloading mahotas-1.4.12-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (
     |████████████████████████████████| 5.7 MB 6.9 MB/s
Requirement already satisfied: imutils in /usr/local/lib/python3.7/dist-packages (from c
Requirement already satisfied: Keras in /usr/local/lib/python3.7/dist-packages (from clo
Collecting commentjson
  Downloading commentjson-0.9.0.tar.gz (8.7 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from clo
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from cloo
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from clo
Requirement already satisfied: progressbar2 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: scikit_learn in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from cl
Collecting lark-parser<0.8.0,>=0.7.1
  Downloading lark-parser-0.7.8.tar.gz (276 kB)
     |████████████████████████████████| 276 kB 42.9 MB/s
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-utils>=2.3.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from progr
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-pac
Building wheels for collected packages: clodsa, commentjson, lark-parser
  Building wheel for clodsa (setup.py) ... done
  Created wheel for clodsa: filename=clodsa-1.2.47-py2.py3-none-any.whl size=74311 sha25
  Stored in directory: /root/.cache/pip/wheels/05/ff/0a/0e6e14c2a68d6869a010e979b8fd9d66
  Building wheel for commentjson (setup.py) ... done
  Created wheel for commentjson: filename=commentjson-0.9.0-py3-none-any.whl size=12093
  Stored in directory: /root/.cache/pip/wheels/eb/bb/07/25a7f0718ee3fe137384011b8e56070
  Building wheel for lark-parser (setup.py) ... done
  Created wheel for lark-parser: filename=lark_parser-0.7.8-py2.py3-none-any.whl size=62
  Stored in directory: /root/.cache/pip/wheels/92/e3/af/1dc0fdca93232d700ac176af6554cf22
Successfully built clodsa commentjson lark-parser
Installing collected packages: lark-parser, mahotas, commentjson, clodsa
Successfully installed clodsa-1.2.47 commentjson-0.9.0 lark-parser-0.7.8 mahotas-1.4.12
```

```
from matplotlib import pyplot as plt
from clodsa.augmentors.augmentorFactory import createAugmentor
from clodsa.transformers.transformerFactory import transformerGenerator
from clodsa.techniques.techniqueFactory import createTechnique
import xml.etree.ElementTree as ET
import cv2
%matplotlib inline
```

```
%cd ..
from google.colab import drive
drive.mount('/content/gdrive')
```

```
     /
     Mounted at /content/gdrive
```

```
!ln -s /content/gdrive/My\ Drive/ /mydrive
```

```
INPUT_PATH = "mydrive/yolov4/camera_micrography"
```

```
PROBLEM = "detection"
ANNOTATION_MODE = "yolo"
GENERATION_MODE = "linear"
OUTPUT_MODE = "yolo"
OUTPUT_PATH= "augmented_images_yolo"
augmentor = createAugmentor(PROBLEM,ANNOTATION_MODE,OUTPUT_MODE,GENERATION_MODE,INPUT_PATH,{"
```

```
img = cv2.imread("mydrive/yolov4/camera_micrography/220.jpg")
# changing to the BGR format of OpenCV to RGB format for matplotlib
plt.imshow(img[:,:,::-1])
```

```
def boxesFromYOLO(imagePath,labelPath):
    image = cv2.imread(imagePath)
    (hI, wI) = image.shape[:2]
    lines = [line.rstrip('\n') for line in open(labelPath)]
    #if(len(objects)<1):
    #    raise Exception("The xml should contain at least one object")
    boxes = []
    if lines != ['']:
        for line in lines:
            components = line.split(" ")
            category = components[0]
            x  = int(float(components[1])*wI - float(components[3])*wI/2)
            y = int(float(components[2])*hI - float(components[4])*hI/2)
            h = int(float(components[4])*hI)
            w = int(float(components[3])*wI)
            boxes.append((category, (x, y, w, h)))
    return (image,boxes)


categoriesColors = {11: (255,0,0),14:(0,0,255)}

def showBoxes(image,boxes):
    cloneImg = image.copy()
    for box in boxes:
        if(len(box)==2):
            (category, (x, y, w, h))=box
        else:
            (category, (x, y, w, h),_)=box
        if int(category) in categoriesColors.keys():
            cv2.rectangle(cloneImg,(x,y),(x+w,y+h),categoriesColors[int(category)],5)
        else:
            cv2.rectangle(cloneImg,(x,y),(x+w,y+h),(0,255,0),5)
    plt.imshow(cloneImg[:,:,::-1])


img,boxes = boxesFromYOLO("mydrive/yolov4/camera_micrography/1.jpg","mydrive/yolov4/camera_mi
showBoxes(img,boxes)
```

```
transformer = transformerGenerator(PROBLEM)


vFlip = createTechnique("flip",{"flip":0})
augmentor.addTransformer(transformer(vFlip))


plt.figure()
plt.title("Original")
showBoxes(img,boxes)
vFlipGenerator = transformer(vFlip)
vFlipImg,vFlipBoxes = vFlipGenerator.transform(img,boxes)
plt.figure()
plt.title("Transformed")
showBoxes(vFlipImg,vFlipBoxes)
```

```
hFlip = createTechnique("flip",{"flip":1})
augmentor.addTransformer(transformer(hFlip))


plt.figure()
plt.title("Original")
showBoxes(img,boxes)
hFlipGenerator = transformer(hFlip)
hFlipImg,hFlipBoxes = hFlipGenerator.transform(img,boxes)
plt.figure()
plt.title("Transformed")
showBoxes(hFlipImg,hFlipBoxes)
```

```
hvFlip = createTechnique("flip",{"flip":-1})
augmentor.addTransformer(transformer(hvFlip))
```

```
plt.figure()
plt.title("Original")
showBoxes(img,boxes)
hvFlipGenerator = transformer(hvFlip)
hvFlipImg,hvFlipBoxes = hvFlipGenerator.transform(img,boxes)
plt.figure()
plt.title("Transformed")
showBoxes(hvFlipImg,hvFlipBoxes)
```

```
rotate = createTechnique("rotate", {"angle" : 90})
augmentor.addTransformer(transformer(rotate))
```

```
plt.figure()
plt.title("Original")
showBoxes(img,boxes)
rotateGenerator = transformer(rotate)
rotateImg,rotateBoxes = rotateGenerator.transform(img,boxes)
plt.figure()
plt.title("Transformed")
showBoxes(rotateImg,rotateBoxes)
```

```
avgBlur =  createTechnique("average_blurring", {"kernel" : 5})
augmentor.addTransformer(transformer(avgBlur))


plt.figure()
plt.title("Original")
showBoxes(img,boxes)
avgBlurGenerator = transformer(avgBlur)
avgBlurImg,avgBlurBoxes = avgBlurGenerator.transform(img,boxes)
plt.figure()
plt.title("Transformed")
showBoxes(avgBlurImg,avgBlurBoxes)
```

```
hue = createTechnique("raise_hue", {"power" : 0.9})
augmentor.addTransformer(transformer(hue))


plt.figure()
plt.title("Original")
showBoxes(img,boxes)
hueGenerator = transformer(hue)
hueImg,hueBoxes = hueGenerator.transform(img,boxes)
plt.figure()
plt.title("Transformed")
showBoxes(hueImg,hueBoxes)
```

```
none = createTechnique("none",{})
augmentor.addTransformer(transformer(none))


augmentor.applyAugmentation()
```

I am executing this notebook in Colaboratory, so I downloaded the generated files.

```
print("Number of images in the folder")
!ls -1 augmented_images_yolo/*.jpg | wc -l
print("Number of annotations in the folder")
!ls -1 augmented_images_yolo/*.txt | wc -l
```

```
     Number of images in the folder
     1631
     Number of annotations in the folder
     1631
```

```
!zip -r augmented_images_yolo.zip augmented_images_yolo
from google.colab import files
files.download('augmented_images_yolo.zip')
```