CV — 目标检测：letterbox

pentiumCM 2021-07-19 2,390 阅读3分钟 专栏： CV

CV — 目标检测：letterbox一、相关概念二、代码实现(一) python代码

- 转载请备注原文出处，谢谢：blog.csdn.net/pentiumCM/a...

CV — 目标检测：letterbox

一、相关概念

0. letterbox：

    - 概念：

        在大多数目标检测算法中，由于 **卷积核为方形**（不排除卷积核有矩形的情况），所以模型输入图片的尺寸也需要为方形。然而大多数数据集的图片基本上为 **矩形**，直接将图片 resize 到正方形，会导致图片失真，比如细长图片中的物体会变畸形。

        letterbox操作：在对图片进行resize时，保持原图的长宽比进行等比例缩放，当长边 resize 到需要的长度时，短边剩下的部分采用灰色填充。

    - 补充点：

        - 在目标检测领域，对数据集图片进行了 letterbox 操作，同时标注框也需要进行 letterbox 操作。

    - 相关算法：

        在 yolo，ssd 等算法的图片预处理过程中，皆使用了 letterbox 处理。

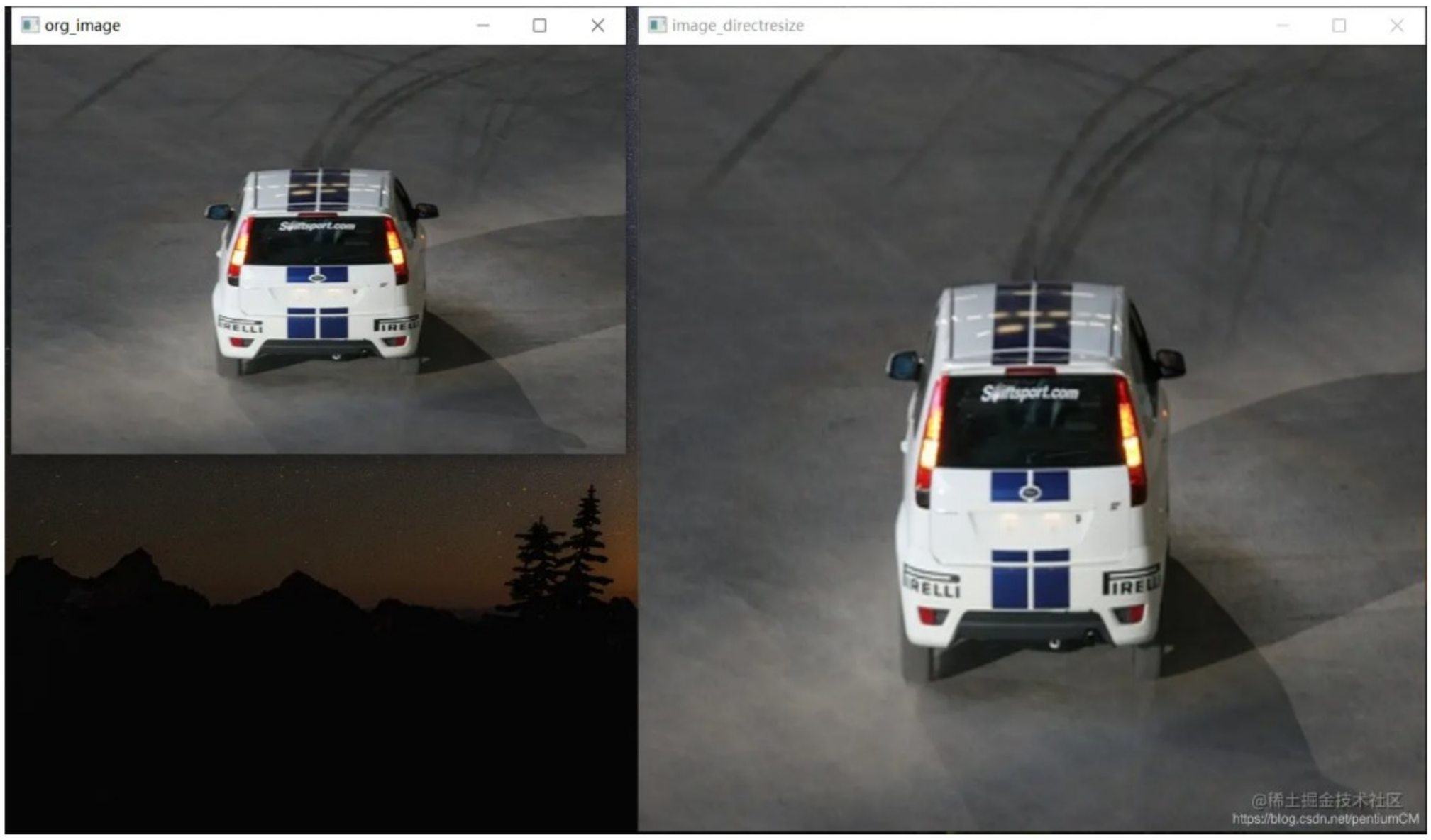
二、代码实现

(一) python代码

0. 样例说明：

- 直接 resize：

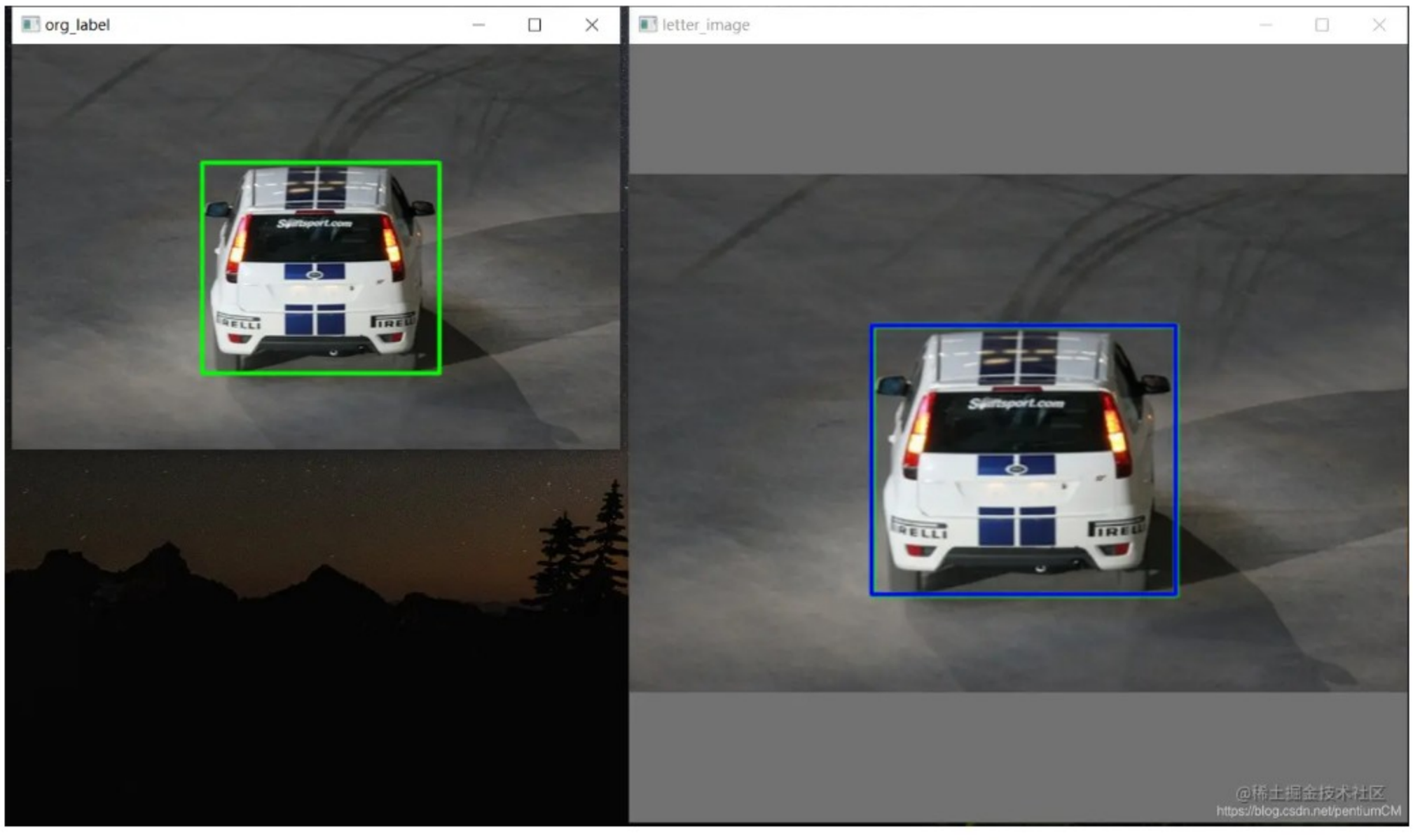
  我们从下图观察，左侧为原图，右侧为直接 resize 之后的图片，明显感觉右侧图片汽车形变失真了



- letterbox 操作：

  右侧图图，我们在进行 resize 时保持了原图的长度比，上下部分不足的部分采用灰色进行填充。

  同时左侧绿色的 标注框 是在原图尺寸下，右侧蓝色是 letterbox 之后的标注框，标注框的坐标也要跟着变换。



1. 完整代码：

ini 复制代码

```ini
#!/usr/bin/env python
# encoding: utf-8
'''
@Author  : pentiumCM
@Email   : 842679178@qq.com
@Software: PyCharm
@File    : util.py
@Time    : 2021/7/17 1:58
@desc    : 目标检测工具类
'''

import cv2
import torch
import numpy as np


def letterbox_image(image_src, dst_size, pad_color=(114, 114, 114)):
    """
    缩放图片，保持长宽比。
    :param image_src:       原图（numpy）
    :param dst_size:        （h，w）
    :param pad_color:       填充颜色，默认是灰色
    :return:
    """
    src_h, src_w = image_src.shape[:2]
    dst_h, dst_w = dst_size
    scale = min(dst_h / src_h, dst_w / src_w)
    pad_h, pad_w = int(round(src_h * scale)), int(round(src_w * scale))

    if image_src.shape[0:2] != (pad_w, pad_h):
        image_dst = cv2.resize(image_src, (pad_w, pad_h), interpolation=cv2.INTER_LINEAR)
    else:
        image_dst = image_src

    top = int((dst_h - pad_h) / 2)
    down = int((dst_h - pad_h + 1) / 2)
    left = int((dst_w - pad_w) / 2)
    right = int((dst_w - pad_w + 1) / 2)

    # add border
    image_dst = cv2.copyMakeBorder(image_dst, top, down, left, right, cv2.BORDER_CONSTANT, value=pa

    x_offset, y_offset = max(left, right) / dst_w, max(top, down) / dst_h
    return image_dst, x_offset, y_offset


def letterbox_label(bounding_box, dst_size=(640, 640), x_offset=0, y_offset=0, normalize=False, src
    """
```

```
    缩放图片，调整 bounding_box 的坐标
    :param bounding_box:    （numpy，（-1，4））标注框，采用归一化的形式，x / w
    :param dst_size:        （tuple）填充之后图片的尺寸，（h，w）
    :param x_offset:        （float）上下填充的大小，归一化形式
    :param y_offset:        （float）左右填充的大小，归一化形式
    :param normalize:       （bool）传入的 bounding_box 是否归一化
    :param src_size:        （tuple）原图的尺寸，（h，w），归一化时候需要
    :return:
    """

    if not normalize:
        assert src_size, 'src_size is None'
        h = src_size[0]
        w = src_size[1]
        bounding_box = bounding_box.astype(np.float)
        bounding_box[:, 0] = bounding_box[:, 0] / w  # top left x
        bounding_box[:, 1] = bounding_box[:, 1] / h  # top left y
        bounding_box[:, 2] = bounding_box[:, 2] / w  # bottom right x
        bounding_box[:, 3] = bounding_box[:, 3] / h  # bottom right y

    y = bounding_box.clone() if isinstance(bounding_box, torch.Tensor) else np.copy(bounding_box)

    # 整体图片尺寸
    pad_h = dst_size[0]
    pad_w = dst_size[1]

    # 内部（除去填充部分）图片尺寸
    inner_w = pad_w * (1 - 2 * x_offset)
    inner_h = pad_h * (1 - 2 * y_offset)

    y[:, 0] = inner_w * bounding_box[:, 0] + pad_w * x_offset  # top left x
    y[:, 1] = inner_h * bounding_box[:, 1] + pad_h * y_offset  # top left y
    y[:, 2] = inner_w * bounding_box[:, 2] + pad_w * x_offset  # bottom right x
    y[:, 3] = inner_h * bounding_box[:, 3] + pad_h * y_offset  # bottom right y

    return y


def plot_one_box(box, image, label=None, color=(0, 255, 0), line_thickness=3):
    """
    Plots one bounding box on image  using OpenCV
    :param box:             bounding_box，xyxy。类型：list
    :param image:
    :param color:
    :param label:
    :param line_thickness:
    :return:
    """

    assert image.data.contiguous, 'Image not contiguous. Apply np.ascontiguousarray(im) to plot_on_
    tl = line_thickness or round(0.002 * (image.shape[0] + image.shape[1]) / 2) + 1  # line/font th

    # 左上，右下
    c1, c2 = (int(box[0]), int(box[1])), (int(box[2]), int(box[3]))
```

```
    cv2.rectangle(image, c1, c2, color, thickness=tl, lineType=cv2.LINE_AA)

    if label:
        tf = max(tl - 1, 1)  # font thickness
        t_size = cv2.getTextSize(label, 0, fontScale=tl / 3, thickness=tf)[0]
        c2 = c1[0] + t_size[0], c1[1] - t_size[1] - 3

        cv2.rectangle(image, c1, c2, color, -1, cv2.LINE_AA)  # filled
        cv2.putText(image, label, (c1[0], c1[1] - 2), 0, tl / 3, [225, 255, 255], thickness=tf, lin


def letterbox_test():
    """
    letterbox 变换测试
    :return:
    """
    # h,w
    dst_size = (640, 640)

    image_path = 'F:/develop_code/python/ssd-pytorch/VOCdevkit/VOC2007/JPEGImages/000012.jpg'
    labels = [156, 97, 351, 270]

    image = cv2.imread(image_path)
    # box: xyxy
    box = np.array(labels, dtype=np.float)
    box = np.reshape(box, (-1, 4))

    cv2.imshow('org_image', image)

    image_directresize = image
    image_directresize = cv2.resize(image_directresize, dst_size)
    cv2.imshow('image_directresize', image_directresize)

    # 可视化标注框
    for i in range(box.shape[0]):
        plot_one_box(box=box[i], image=image, line_thickness=2)

    letter_image, x_offset, y_offset = letterbox_image(image, dst_size)
    letter_box = letterbox_label(box, dst_size, x_offset, y_offset, False, image.shape[:-1])

    for i in range(letter_box.shape[0]):
        plot_one_box(box=letter_box[i], image=letter_image, line_thickness=2, color=(255, 0, 0))

    cv2.imshow('org_label', image)
    cv2.imshow('letter_image', letter_image)

    cv2.waitKey()


if __name__ == '__main__':
    letterbox_test()
```

\