数字图像处理II

学号：201983160037
姓名：强盛周
班级：19信计嵌入1班
邮箱：qshengz@foxmail.com
授课老师：陈允杰教授

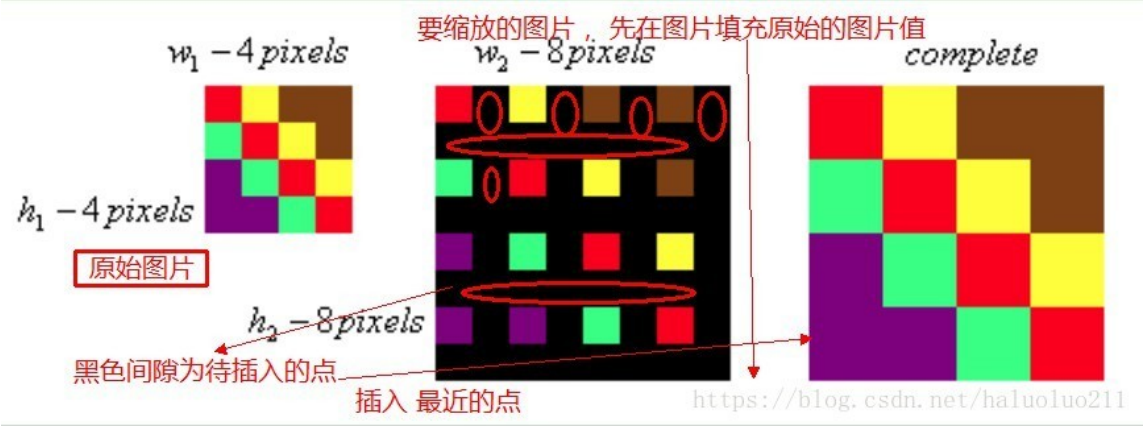
第一次课堂作业

1. 采用最临近值插值方法放大图片2倍，4倍
2. 使用双线性插值方法完成同上
3. 选择不同类别的图片

1. 临近值插值

最简单的图像缩放算法就是最近邻插值。顾名思义，就是将目标图像各点的像素值设为源图像中与其最近的点。算法优点在与简单、速度快。

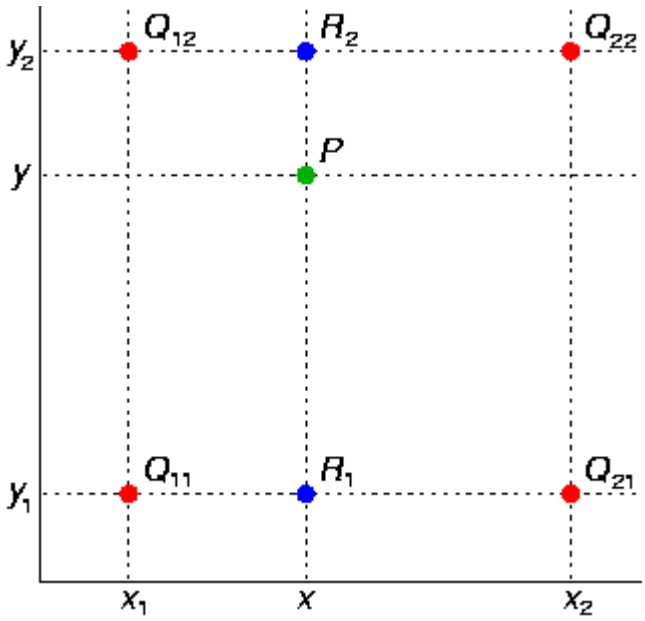
如下图所示，一个*44的图片缩放为*88的图片。步骤：



生成一张空白的8*8的图片，然后在缩放位置填充原始图片值(可以这么理解)

在图片的未填充区域(黑色部分)，填充为原有图片最近的位置的像素值。

2. 双线性插值



$Q_{11}, Q_{12}, Q_{21}, Q_{22}$ 为近邻点， P 为待求点。 设算子 $f(\cdot)$ 代表该像素原图像的灰度值。 Q_{ij} 的坐标为: (x_i, y_j) 则第一次线性插值计算 R_1、 R_2 : (也用 R_1、 R_2 分布代表他们的线性插值得到的灰度值)

$$R_1 = \frac{x_2 - x}{x_2 - x1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$
$$R_2 = \frac{x_2 - x}{x_2 - x1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

第二次插值:

$$P = \frac{y_2 - y}{y_2 - y_1} R_1 + \frac{y - y_1}{y_2 - y1} R_2$$

还记得提到过: $Q_{11}, Q_{12}, Q_{21}, Q_{22}$ 为近邻点吗，所以上面各式的分母 $y_2 - y_1, x_2 - x_1$ 都等于 1 。 于是最终得到:

$$\begin{gathered} P = f(Q_{11})(x_2 - x)(y_2 - y) + \\ f(Q_{21})(x - x_1)(y_2 - y) + \\ f(Q_{12})(x_2 - x)(y - y_1) + \\ f(Q_{22})(x - x_1)(y - y_1) \end{gathered}$$

源图像和目标图像几何中心的对齐 方法: 在计算源图像的虚拟浮点坐标的时候，中心对齐:

$$SrcX = (dstX + 0.5) * (srcWidth / dstWidth) - 0.5$$
$$SrcY = (dstY + 0.5) * (srcHeight / dstHeight) - 0.5$$

参考

1. 数字图像处理笔记二 - 图片缩放(最近邻插值(Nearest Neighbor interpolation)) https://cloud.tencent.com/developer/article/1338324#4.2
2. 图像缩放--双线性内插法及其python实现（图文代码）http://t.csdn.cn/OnpQ2
3. 深入理解双线性插值算法 http://t.csdn.cn/BoiII
4. 维基百科：双线性插值 https://zh.wikipedia.org/wiki/%E5%8F%8C%E7%BA%BF%E6%80%A7%E6%8F%92%E5%80%

In [1]:
```
# @Author: Alephant——QSZ
import numpy as np
import matplotlib.pylab as plt
from matplotlib import image as mpimg
```

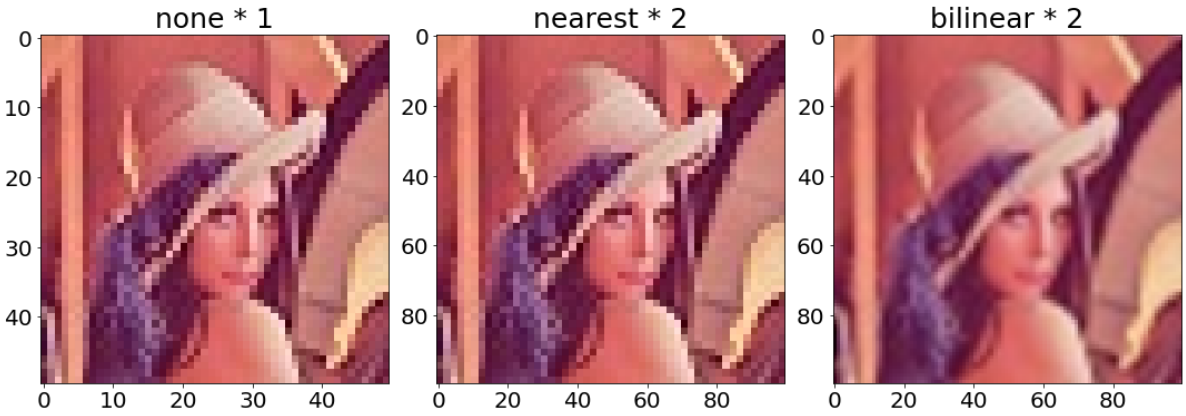
In [2]:
```
def img_resize(img, mag=1, method='nearest'):
    """
    使用插值方法重新改变图片的大小
    :param img: 图片信息，格式为mpimg.imread()
    :param mag: 缩放的倍率
    :param method: 使用的方法，现在有最临近值插值—'nearest'和# 双线性插值—'bilinear'
    :return: 插值后的图片
    """
    mag = float(mag) # 将倍率强制转换为浮点数
    height, width, channels = img.shape# 获取原图的高、宽、通道数
    new_h, new_w = int(height * mag), int(width * mag) # 计算新图的高和宽
    new_img = np.zeros((new_h,new_w,channels)) # 初始化新图的张量

    # 最临近值插值
    if method == 'nearest':
        # 遍历每个新图的像素点，在这里高坐标代表y轴坐标，宽坐标代表x轴坐标
        for new_h_y in range(new_h):
            for new_h_x in range(new_w):
                # 原来的宽、高坐标与新坐标的关系是有一个倍率的整数
                w_x = int(new_h_x / mag)
                h_y = int(new_h_y / mag)
                # 在新图该位置赋值
                new_img[new_h_y, new_h_x] = img[h_y, w_x]
        return new_img


    # 双线性插值
    elif method == 'bilinear':
        # 遍历每个新图的像素点
        for new_h_y in range(new_h):
            for new_h_x in range(new_w):
                # 将每个像素看做正方形，进行中心对齐，投影目标图的横轴和纵轴到原图上
                w_x = (new_h_x + 0.5) / mag - 0.5
                h_y = (new_h_y +0.5) / mag - 0.5

                # 找出每个投影点在原图横轴方向的近邻点坐标
                # 原图范围是0~width-2, 0~height-2是因为要考虑+1后的边界
                f_x = np.clip(w_x, 0, width-2).astype(int)
                f_y = np.clip(h_y, 0, height-2).astype(int)

                # 给出原轴的4个最近邻点Q坐标
                x1, x2 = f_x, f_x+1
                y1, y2 = f_y, f_y+1

                # 给出这四个Q点对应的权重，注意图像数据中y轴在第一位，x轴在第二位
                f_Q11 = img[y1, x1]
                f_Q12 = img[y2, x1]
                f_Q21 = img[y1, x2]
                f_Q22 = img[y2, x2]

                # 计算两个R点对应的权重
                f_R1 = (x2 - w_x) / (x2 - x1)*f_Q11 + (w_x - x1) / (x2 - x1)*f_Q21
                f_R2 = (x2 - w_x) / (x2 - x1)*f_Q12 + (w_x - x1) / (x2 - x1)*f_Q22

                # 计算投影点P的双线性插值，将其赋值于新图该点像素
                f_P = (y2 - h_y) / (y2 - y1)*f_R1 + (h_y - y1) / (y2 - y1)*f_R2
```

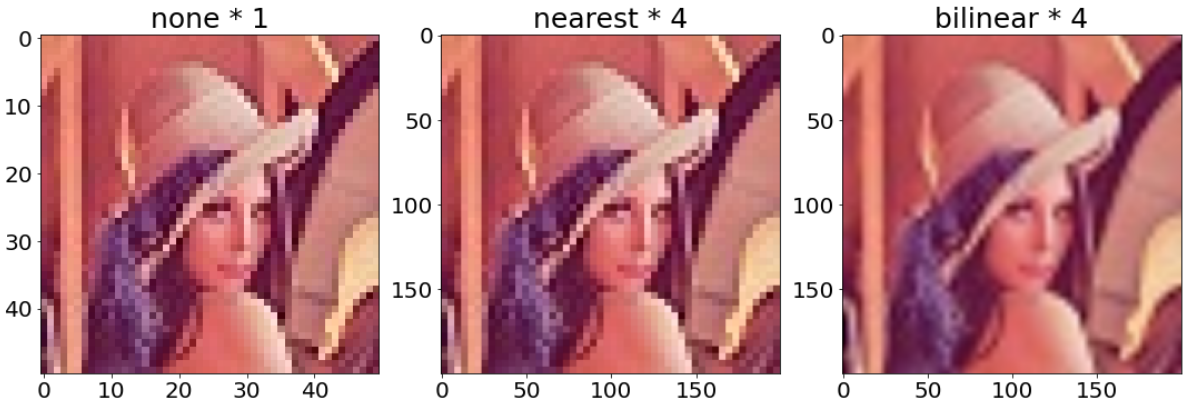
```
                new_img[new_h_y, new_h_x] = f_P
        return new_img
    else:
        return img
```

In [3]:
```
def compare_method(path, mags=np.array([1,2,2])):
    """
    封装了对比图片的方法，只需要输入图片地址和倍率即可生成对比图
    :param path: 图片地址
    :param mags: 放大倍率，默认第一个值为1
    :return: 一个一行三列的子图图片
    """
    # 读取图片
    im = mpimg.imread(path)
    methods = ['none', 'nearest','bilinear']
    fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 30))
    # 无坐标轴代码
    # fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 30), subplot_kw={'xt
    fig.subplots_adjust(hspace=0.05, wspace=0.05)
    for ax, mag, interp_method in zip(axes.flat, mags, methods):
        new_im = img_resize(im, mag, interp_method)
        ax.imshow(new_im)
        ax.set_title(str(interp_method)+" * "+str(mag), size=25)
        ax.tick_params(labelsize=20)
    plt.tight_layout()
    # plt.show()
```

In [4]:
```
# 使用不同图片对比最近邻插值和双线性插值法
# 使用默认图片，放大两倍
compare_method(path="../images/lena_smell.png") # 使用上级images文件中照片
plt.savefig("./pic/lena_smell_p.png", bbox_inches='tight') # 将p过的图存放在本文件夹

# 放大四倍
compare_method(path="../images/lena_smell.png", mags=[1, 4, 4])
plt.savefig("./pic/lena_smell_p_4.png", bbox_inches='tight')
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats
or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats
or [0..255] for integers).
```



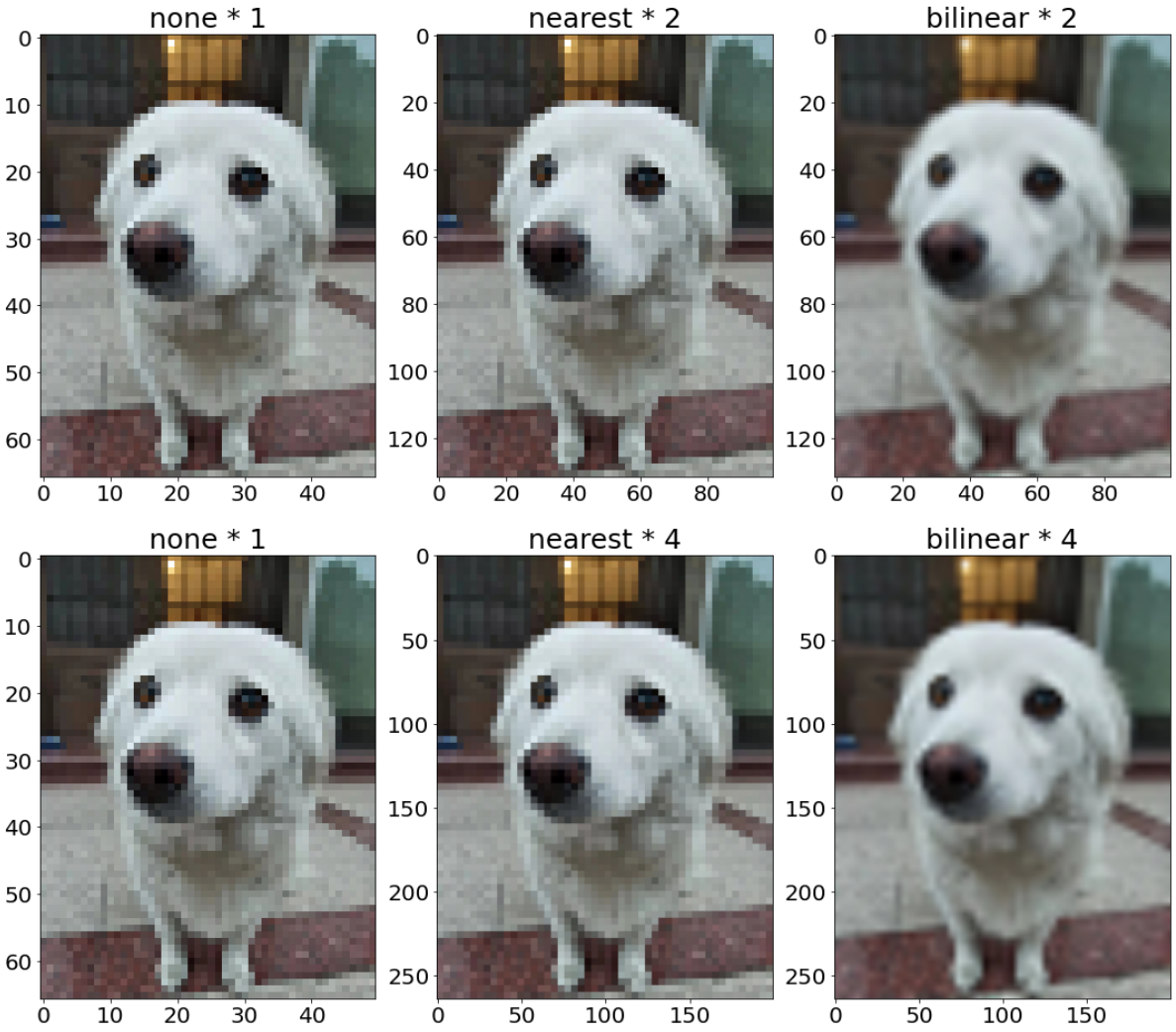


In [5]:
```
# zyw——人像
compare_method(path="../images/zyw.png")
plt.savefig("./pic/zyw_p.png", bbox_inches='tight')

# 放大四倍
compare_method(path="../images/zyw.png", mags=[1, 4, 4])
plt.savefig("./pic/zyw_p_4.png", bbox_inches='tight')
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

In [6]:
```
# 狗子——动物
compare_method(path="../images/dog.png")
plt.savefig("./pic/dog_p.png", bbox_inches='tight')

# 放大四倍
```

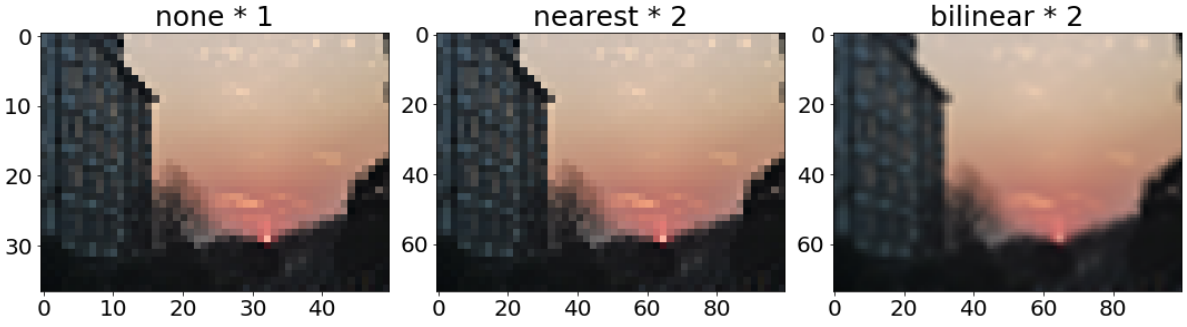
```
compare_method(path="../images/dog.png", mags=[1, 4, 4])
plt.savefig("./pic/dog_p_4.png", bbox_inches='tight')
```

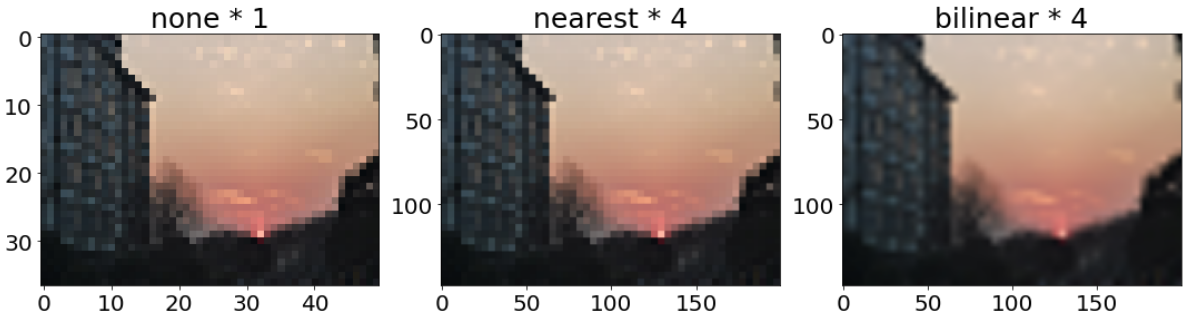


In [7]:

```
# 日落—风景
compare_method(path="../images/sunset.png")
plt.savefig("./pic/sunset_p.png", bbox_inches='tight')

# 放大四倍
compare_method(path="../images/sunset.png", mags=[1, 4, 4])
plt.savefig("./pic/sunset_p_4.png", bbox_inches='tight')
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```





In [8]:
```
# 大树—风景
compare_method(path="../images/tree.png")
plt.savefig("./pic/tree_p.png", bbox_inches='tight')

# 放大四倍
compare_method(path="../images/tree.png", mags=[1, 4, 4])
plt.savefig("./pic/tree_p_4.png", bbox_inches='tight')
```