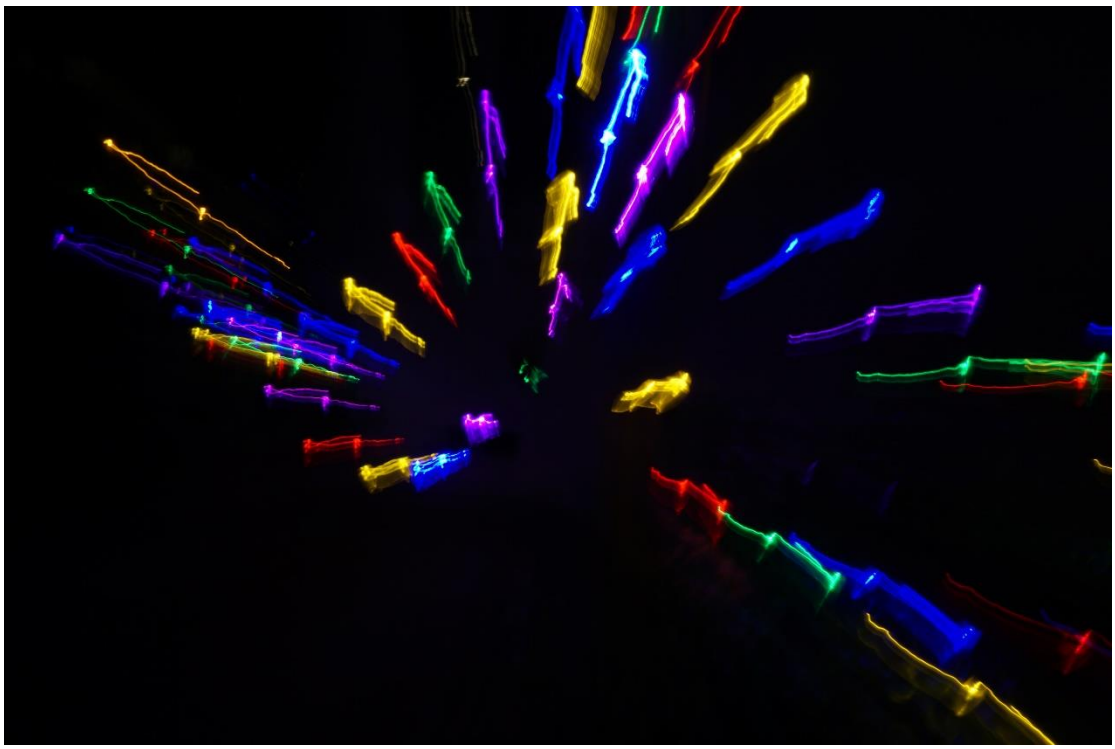
动感光影和马赛克报告

Python 实现旋转，缩放，平移

原理：对原图进行旋转，缩放，平移，高斯模糊以及多种组合，最后进行亮度优先叠加

效果图：

原图：



缩放然后叠加：

平移加缩放再叠加：

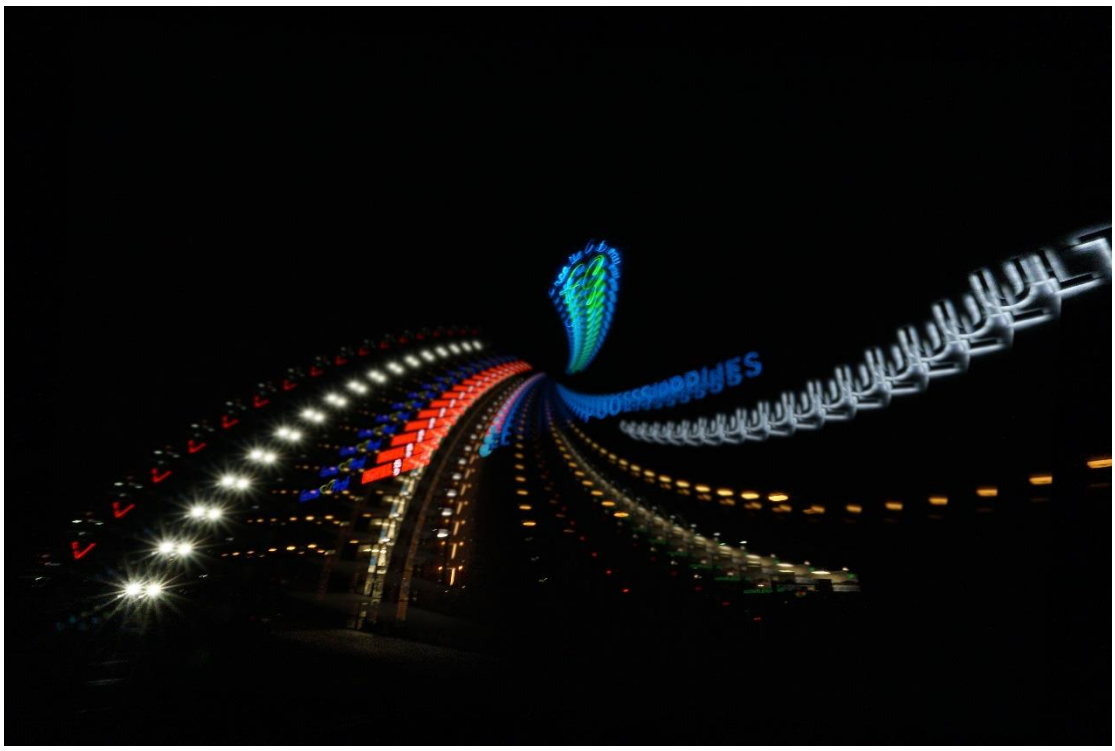
旋转叠加：

旋转缩放叠加：

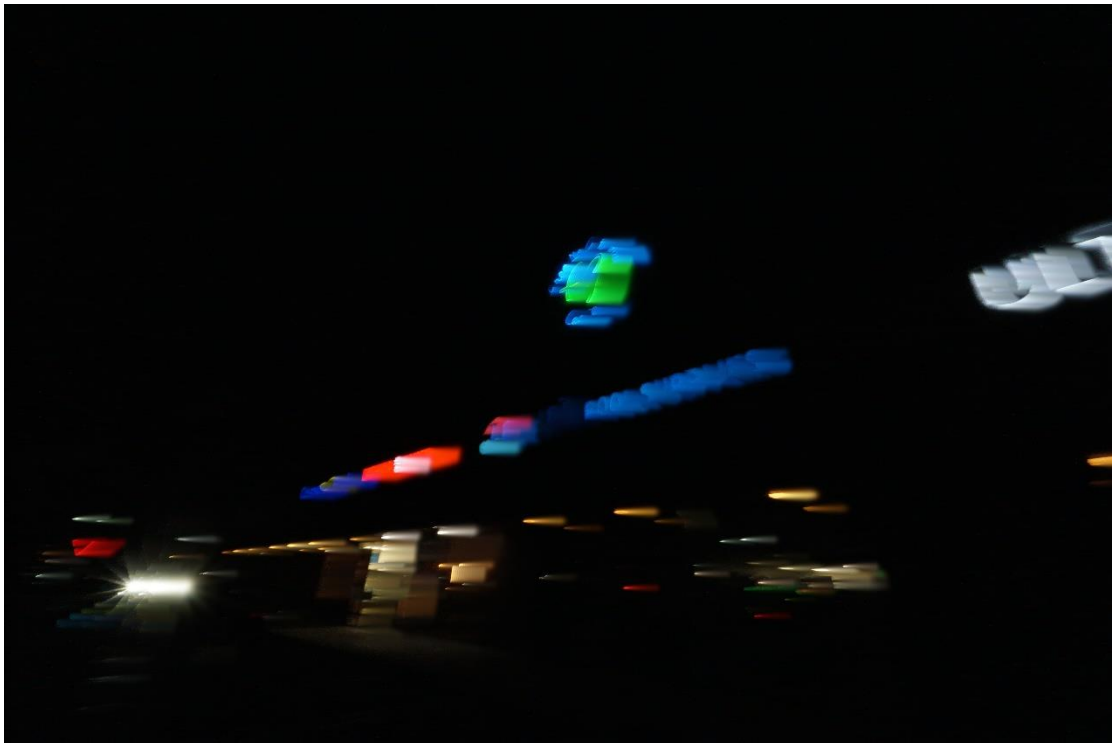
平移加高斯模糊：

原图：

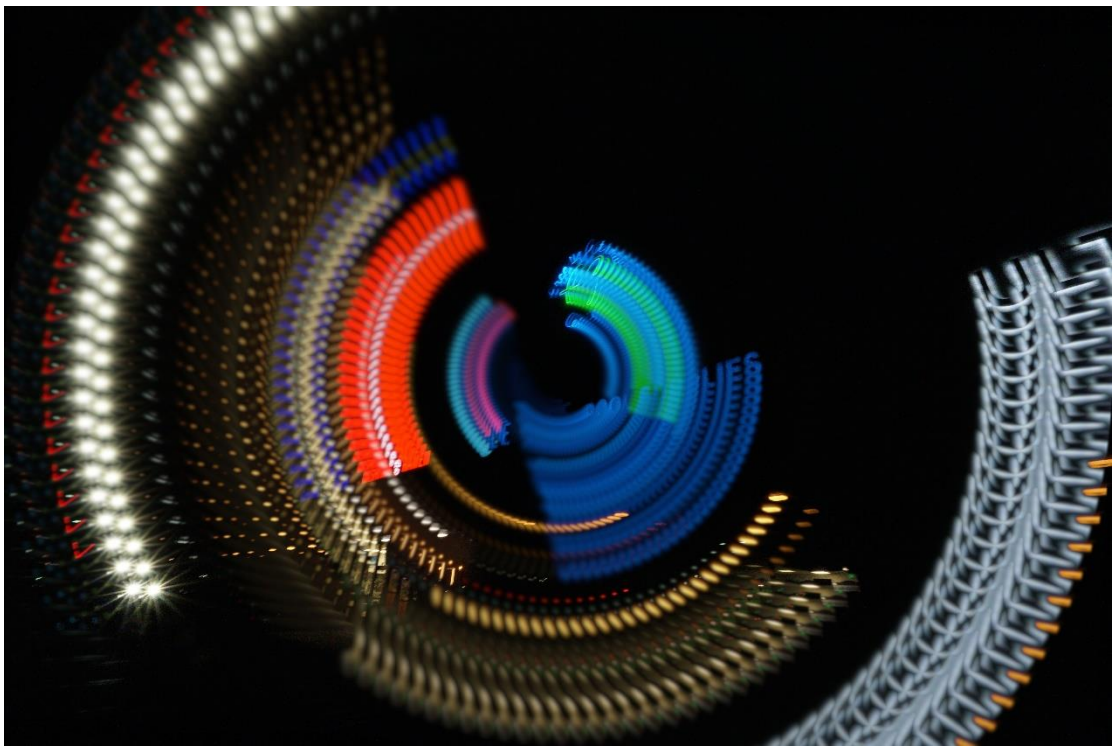
旋转缩放：



平移加高斯模糊：



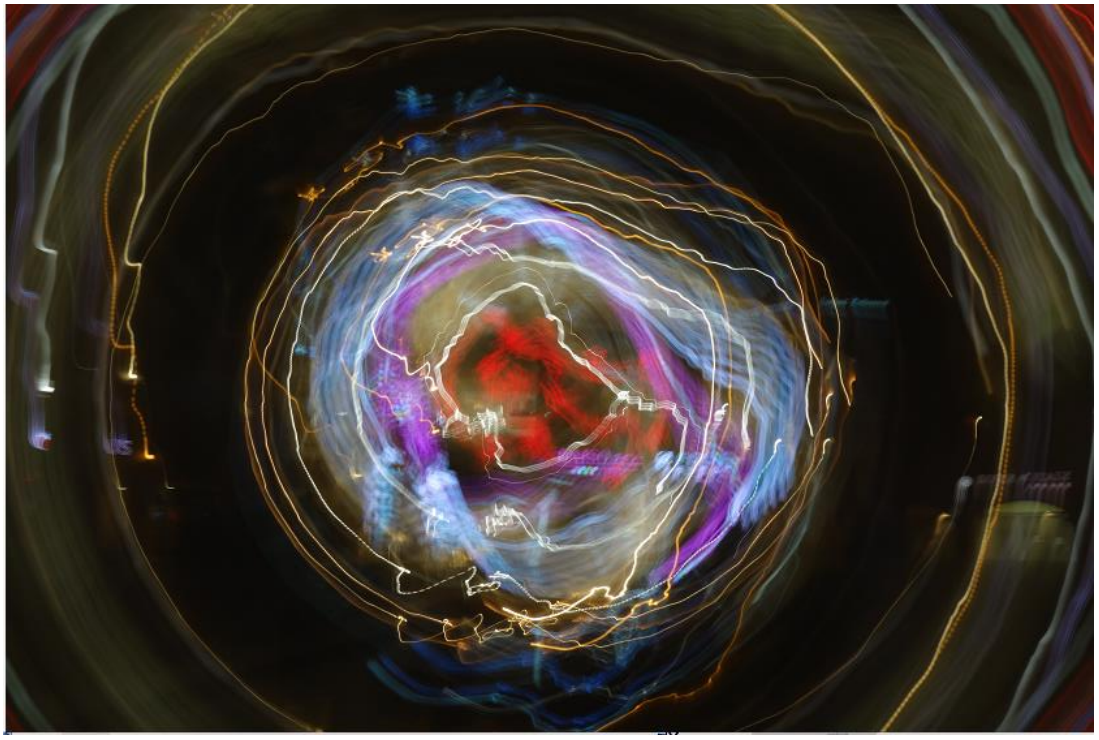
旋转加高斯模糊：



存在问题：

1 没有实现老师发的效果图那种不规则运动拉丝的效果

2 下图中的模糊与清晰相间的效果不知道如何实现



下一步的计划：

尝试曲线运动（圆锥曲线等）

完整的 Python 代码：

```python
import numpy as np
import cv2


def get_rotate_scale(center, step_angle, step_scale=1):
    """
    :param center: 旋转中心
    :param step_angle: 旋转角度
    :param step_scale: 缩放因子
    :return: 旋转+缩放变换矩阵
    """
    r = cv2.getRotationMatrix2D(center, step_angle, step_scale)
    return r


def get_translation(step_x, step_y):
    """
    :param step_x: 水平移动距离
    :param step_y: 垂直移动距离
    :return: 平移变换矩阵
    """
    t = [[1, 0, step_x], [0, 1, step_y]]
    t = np.array(t, dtype=np.float32)
    return t


def get_trans_img(frames, original_img, t, blur_type='',
kneral_size=(5, 5)):
    """

    :param frames: 结果图像帧数
    :param original_img: 原始图像
    :param t: 几何变换
    :param blur_type: 滤波类型
    :param kneral_size: 滤波尺寸
    :return:
    """
    rows, cols, channels = original_img.shape
    res = [original_img]
    pre_img = original_img
    for i in range(frames):
        cur_img = cv2.warpAffine(pre_img, t, (cols, rows))
        if blur_type == 'Gauss':
            cur_img = cv2.GaussianBlur(cur_img, kneral_size, 0)
```

```
        pre_img = cur_img
        res.append(cur_img)
    return res


def trans_merge(t1, t2):
    """
    为了加速运算
    :param t1: 变换 1 矩阵
    :param t2: 变换 2 矩阵
    :return: 融合变换矩阵
    """
    transform = []
    for i in range(len(t1)):
        # opencv 使用齐次坐标变换，保证变换矩阵能混合，除第一个矩阵，其余矩阵加一行[0, 0, 1]
        temp = np.concatenate((t2[i], np.array([[0, 0, 1]])), axis=0)
        transform.append(np.dot(t1[i], temp))
    return transform


def img_merge(list_img):
    """

    :param list_img: 图像列表
    :return: 融合后的图像-亮度优先
    """
    res = list_img[0]
    num = 0
    for img in list_img:
        cv2.imwrite('move' + str(num) + '.jpg', img)
        num += 1

        for i in range(img.shape[0]):
            for j in range(img.shape[1]):
                res[i, j, 0] = max(res[i, j, 0], img[i, j, 0])
                res[i, j, 1] = max(res[i, j, 1], img[i, j, 1])
                res[i, j, 2] = max(res[i, j, 2], img[i, j, 2])
    return res


if __name__ == "__main__":
    img = cv2.imread('1.jpg')
    rows, cols, channels = img.shape
```

```
    # 几何变换
    # rotate = get_rotate_scale((cols/2, rows/2), -3, 0.9)  # 旋转+缩小
    translation = get_translation(10, 0)  # 平移
    # scale = get_rotate_scale(10, (cols/2, rows/2), 0, 0.9)  # 使用旋转的缩放因子，但旋转角度为 0
    # trans = trans_merge(scale, translation)  # 变换融合
    result = get_trans_img(16, img, translation, 'Gauss', (21, 21))  # 变换后的帧图像

    # 图像叠加-亮度优先
    res = img_merge(result)

    cv2.imwrite('move45.jpg', res)
    cv2.waitKey()
```

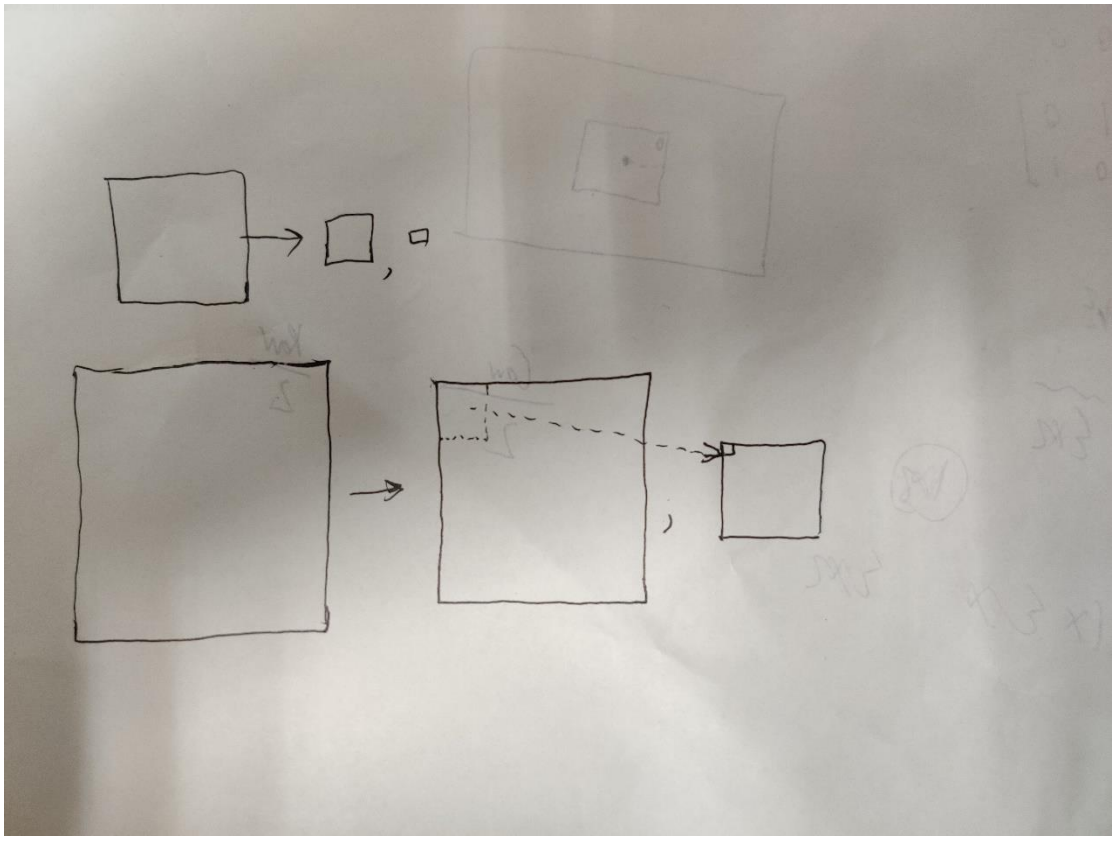
马赛克设计思路

1. 丰富照片库:
   对原照片做对比度拉伸，均值方差规定化，直方图均衡化，对亮度，饱和度进行调整，丰富图库，完成颜色的扩充
2. 照片预处理:
   ① 对小照片进行人脸目标检测和视觉显著性分割找到一幅照片的主体
   ② 对前景图做插值，使大图变大（使小图和前景图的小块像素量一致），纹理区域变得很细（对眉毛胡子区域有意义）
3. 照片匹配:

有纹理部分做纹理匹配（感知哈希算法），无纹理部分做颜色匹配（颜色分布直方图）

老师给的 mosaic.py 代码的处理步骤：

1 每个小图片处理得到设定小图片尺寸的图片和特征（灰度值均值），底片处理得到尺寸是设定小图片尺寸整数倍的图片和降采样后的图片（每个像素的灰度值是原图像中设定小图片尺寸区域的灰度值均值）。
2 对底片每块区域，用降采样后的对应位置的灰度值取匹配小图片的特征中欧氏距离最小的，作为最佳匹配项，记录下标
3 对底片每块区域，按上步得到的小图片下标索引到小图片，然后粘贴。



可改进的地方（小图片尺寸一致）：

1 代码使用的特征是小图片的灰度均值，这样小图片的颜色信息丢失严重，可以考虑先使用感知哈希法（64 位整数哈希值）、颜色分步法-直方图（64 维向量），作为特征。
2 代码将小图片降采样来到达设定的尺寸，小图片的质量较差，实现上改进为对小照片进行截取，同时这里截取需要 AI 算法的加入，人像等不能截取部分。
3 最佳匹配需根据选择的小图片的特征类型进行选择，如感知哈希法-汉明距离
4 透明度线性融合
5 后续可以考虑小图片不同尺寸，其次 AI 算法的实现优化，再者以上都是基于颜色匹配，可以考虑加入内容对应即某个塔尖就是由多幅小图片中的内容构建而成。