Otsu thresholding algorithm by python

目录

1.工程概述

1.1 编写目的

使用 python 实现 otsu，并对其进行优化。
优化方式包括：

1. 使用灰度直方图减少遍历迭代次数。
2. 使用遗传算法和退火算法进行快速收敛。

1.2 所需依赖

1. python3
2. numpy 库（import numpy）
3. python-imaging 库（简称 PIL，import PIL）
4. time 库（import time）

1.3 如何使用

1. 克隆这个工程。然后运行 "use_hist.py" 或 "GA_Main.py"。
2. 如果你运行了"use_hist.py",你将得到使用灰度直方图计算的 otsu 分割阈值。
3. 如果你运行了"GA_Main.py"，你将得到使用遗传算法和退火算法优化计算得出的 otsu 分割阈值。
4. 务必传入正确的图像地址字符串以运行程序。
5. 本工程只接受处理彩色图片。

2.相关算法简述

2.1Otsu

Otsu 算法中，使用穷举法找到能使类内方差最小的阈值，最小类内方差的计算公式为：

$$\sigma^2{}_{\omega}(t)=\omega_1(t)\sigma^2{}_1(t)+\omega_2(t)\sigma^2{}_2(t)$$

公式中，t 为阈值，ω_i 是被阈值分开的两个类的概率（即前景和背景的像素点占比），$\sigma^2{}_i(t)$ 则为两个类的方差。
Otsu 以证明，最小化的类内方差和最大化的类间方差是相同的，最大类间方差的计算公式

为：

$$\sigma^2{}_{\mathrm{b}}(\mathrm{t})=\sigma^2\text{-}\sigma^2{}_{\omega}(\mathrm{t})=\omega_1(\mathrm{t})\omega_2(\mathrm{t})\left[\mu_1(\mathrm{t})\text{-}\mu_2(\mathrm{t})\right]^2$$

同样，公式中 t 为阈值，ω_{i} 表示类概率（即前景和背景的像素点占比），μ_{i} 表示类均值（即前景和背景的像素点平均值）。

2.2 使用灰度直方图的 otsu

2.2.1 参数简述

$\omega_1(\mathrm{t})$ 前景类概率（即前景像素点占比）可以通过直方图计算。

$$\omega_1(\mathrm{t})=\sum_{\mathrm{i}=0}^{\mathrm{t}}\mathrm{p}(i)[0\le i\le 255\,\&\,i\in N]$$

其中 $p(i)$ 为相应像素值出现的概率（即某一像素值的占比），$p(i)$ 可由灰度直方图得出。

显而易见，$\omega_2(\mathrm{t})=1\text{-}\omega_1(\mathrm{t})$

$\mu_1(\mathrm{t})$ 前景均值（即前景像素点平均值）为：

$$\mu_1(\mathrm{t})=\sum_{\mathrm{i}=0}^{\mathrm{t}}i\cdot n\cdot p(i)\div\sum_{i=0}^{\mathrm{t}}n\cdot p(i)[0\le i\le 255\,\&\,i\in N]$$

其中 n 为总像素点个数。

μ 为总像素均值，计算公式为：

$$\mu=\sum_{\mathrm{i}=0}^{255}i\cdot n\cdot p(i)[0\le i\le 255\,\&\,i\in N]$$

$\mu_2(\mathrm{t})$ 背景均值（即背景像素点平均值）为：

$$\mu_2=(\mu-\omega_1\cdot\mu_1)\div\omega_2$$

2.2.2 算法简述

1. 计算每个强度级的直方图和概率。
2. 通过直方图得到上文描述的相关参数。
3. 将相关参数代入最大类间方差的计算公式

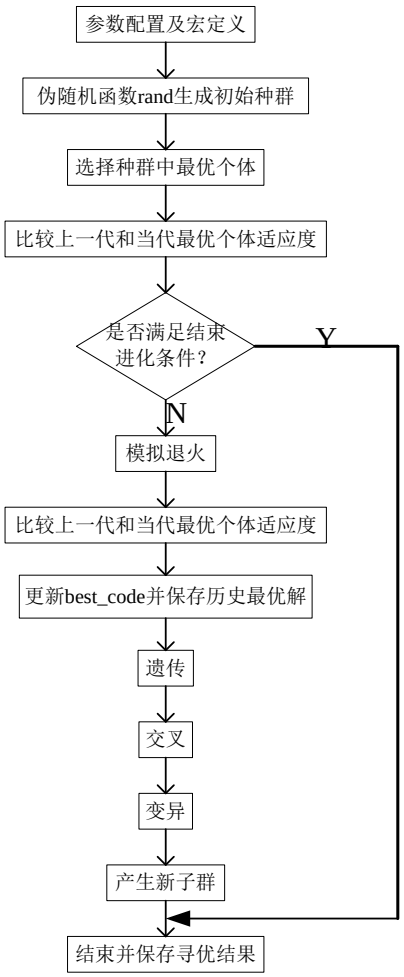
$$\sigma^2{}_{\mathrm{b}}(\mathrm{t})=\sigma^2\text{-}\sigma^2{}_{\omega}(\mathrm{t})=\omega_1(\mathrm{t})\omega_2(\mathrm{t})\left[\mu_1(\mathrm{t})\text{-}\mu_2(\mathrm{t})\right]^2$$

4. 遍历所有可能的阈值 t，$0 \le t \le 255 \& t \in N$
5. 得到最大类间方差所对应的阈值 t

2.3 使用遗传算法和退火算法优化的 otsu

2.3.1 遗传退火算法简述

1. 生成一代初始种群，这个过程是随机的，但是长度、规模、概率都是事先给出的固定值。
2. 对初代种群计算自适应度。
3. 判断是否符合终止条件，如果不满足，就进行选择、交叉、变异等操作生成下一代新的种群，跳到上一步进行重复循环，如果满足就直接输出结果。
4. 比较两代最优个体，根据退火概率有选择的保留 best_code。
5. 输出结果。



2.3.2 相关参数简述

需要寻优的函数为 otsu 的最大类间方差函数：

$$\sigma^2{}_{\mathrm{b}}(\mathrm{t})=\sigma^2\text{-}\sigma^2{}_{\omega}(\mathrm{t})=\omega_1(\mathrm{t})\omega_2(\mathrm{t})\left[\mu_1(\mathrm{t})\text{-}\mu_2(\mathrm{t})\right]^2$$

初始化设定种群大小。
根据种群大小随机生成相应个数的个体。

种群个体为像素值 t，$0 \le t \le 255 \& t \in N$ 。

使用 otsu 的最大类间方差函数作为适应度函数：

适应度函数：$\sigma^2{}_{\mathrm{b}}(\mathrm{t})=\sigma^2\text{-}\sigma^2{}_{\omega}(\mathrm{t})=\omega_1(\mathrm{t})\omega_2(\mathrm{t})\left[\mu_1(\mathrm{t})\text{-}\mu_2(\mathrm{t})\right]^2$

将 otsu 的最大类间方差函数作乘以-1，作为不适应度函数：

不适应度函数：$\sigma^2{}_b(t) = -1\cdot\omega_1(t)\omega_2(t)[\mu_1(t)-\mu_2(t)]^2$

将种群个体进行基因编码，及将 10 进制的 t，转化为 8 位 2 进制数。
设定相应的复制概率，交叉概率，编译概率，以及最大遗传代数。
经过相应次数的遗传与退火，保留相对优秀的个体。
最后保留的个体即为需要寻找的阈值。
将 8 位 2 进制编码转化为 10 进制数，得到最终结果。

3.相关文件描述

3.1 use_hist.py

使用灰度直方图计算 otsu 的分割阈值。
传入参数：pic_address = “图片地址的字符串”
结果：最大类间方差
Otsu 阈值
程序耗时
输出图像

3.2 GA_Main.py

使用遗传退火优化 otsu 的主函数。
可调参数：popsize = 种群大小
codelength = 种群个体编码长度
max_GN_limtation = 允许进化的最大代数
ps = 复制概率
pc = 交叉概率

pm = 变异概率
结果：otsu 阈值
程序耗时

3.3 used_function.py

传入参数：pic_address = “图片地址的字符串”
遗传算法中使用到的函数：
目标函数：
obj_funtion()
适应度函数：
fit_funtion()
不适应度函数：
ufit_funtion()

3.4 Evaluate.py

遗传算法评价函数：
接收参数：pop（一个种群列表或一维矩阵）
返回值：该种群的最优个体的编码。

3.5 Select.py

遗传算法复制函数：
接收参数：pop（一个种群列表或一维矩阵），ps 复制概率
返回值：复制了优秀个体并替换了劣质个体后的种群。
涉及算法：轮盘赌算法。

3.6 Crossover.py

遗传算法交叉函数：
接收参数：pop（一个种群列表或一维矩阵），pc 交叉概率
返回值：相应个体进行交叉后的种群。
交叉实现：相应个体编码与掩码按位与。

3.7 Mutation.py

遗传算法变异函数：
接收参数：pop（一个种群列表或一维矩阵），pm 交叉概率
返回值：相应个体进行变异后的种群。
变异实现：相应个体编码与随机数按位或。

4.出错及容错处理

4.1 文件"Select.py"中的错误

错误：
在对种群个体进行 0-1 量纲化时，使用到函数：

$$i = \frac{i - \min}{\max - \min}$$

其中 i 为当前个体，max 为种群最优个体，min 为种群最劣个体。
如果种群个体都一样，则 0-1 量纲化出错。
解决方案：

使用 python 自身的容错机制，使程序不中断。

并人为抛弃这一代种群。

5.测试结果

5.1 测试程序："use_hist.py"

测试次数	测试图片	阈值	运行时间
1	lena512color.jpg	116	0.46257472038269043 s
2	lena512color.jpg	116	0.44468045234680176 s
3	lena512color.jpg	116	0.44128990173339844 s

5.2 测试程序："GA_Main.py"

测试次数	测试图片	阈值	运行时间
1	lena512color.jpg	116	0.4816863536834717 s
2	lena512color.jpg	118	0.5131931304931641 s
3	lena512color.jpg	116	0.4674417972564697 s

5.3 对猪养殖过程相关图像的测试