Project 2

数字水印检测系统的实现，基于数字水印的图片泄露检测，编程实现图片水印嵌入和提取（可依托开源项目二次开发），并进行鲁棒性测试，包括不限于翻转、平移、截取、调对比度等。

代码核心是一个 CustomWatermarkDetector 类，包含水印生成、嵌入、变换、检测和鲁棒性测试等功能。整体流程分为：

1、初始化水印参数（文字、字体、大小、颜色等）

2、生成水印图案

3、将水印嵌入原图

4、对含水印的图片施加各种变换（模拟图片被篡改的场景）

5、检测变换后图片中的水印，计算与原始水印的相似度

6、输出测试结果

一、初始化方法

```
def __init__(self, watermark_text="Confidential",
             seed=42,font_path=None,
             font_size_ratio=0.1,
             text_color=(255, 255, 255, 30)):
```

设置水印的核心参数决定水印的样式和分布规律

Watermark_text:水印文字内容

seed：随机种子，保证水印分布的一致性（相同种子生成的水印位置相同）

font_path:字体文件路径

font_size_radtio:字体大小与图片最小边的比例

Text_color:水印颜色（RGBA 格式，前 3 位是红/绿/蓝，第 4 位是透明度，0-255 之间，值越小越透明）。

二、文本尺寸计算

```
def get_text_size(self, font, text):
```

兼容不同版本的 Pillow 库，获取文本的宽高尺寸。

三、生成水印

```
def generate_text_watermark(self, size):
```

根据设置的参数生成透明背景的水印图案。

1、创建透明背景的空白图像（RGBA 模式，透明通道全为 0）。

2、加载字体：优先用 font_path 指定的字体，失败则用系统默认字体。

3、计算字体大小：根据图片尺寸和 fron_size_radtio 确定。

4、计算文本尺寸：用 get_text_size 方法获取文字的宽高。

5、随机分布水印：在图片上均匀分布多个文字，每个文字位置随机偏移、旋转角度随机。

6、合成水印：将所有文字绘制到透明背景上，形成最终水印图案。

四、嵌入水印

```
def embed_watermark(self,image_path,
                    output_path=None):
```

将生成的水印嵌入到原始图片中，并保存结果。

1、打开原始图片并转换为 RGBA 模式（支持透明通道）

2、调用 generate_text_watermark 生成与原图尺寸一致的水印。

3、合并原图和水印：用 Image.alpha_composite 叠加

4、转换为 RGB 模式（去掉透明通道，方便保存为 JPG 等格式）并保存（如果指定了 output_path）。

五、图像变换

```
def apply_transformations(self, image,
                          transform_type):
```

对含水印的图片施加各种变换（模拟图片被编辑、篡改的场景），用于测试水印的抗干扰能力。

变换类型有水平/垂直翻转、旋转、裁剪、缩放、亮度调整、对比度调整。

六、水印检测

```
def detect_watermark(self, image,
                     original_watermark):
```

检测变换后的图片中水印的保留程度，计算与原始水印的相似度（百分比）。

1、将变换后的图片和原始水印都转换为 RGBA 模式。

2、调整水印尺寸以匹配图片尺寸。

3、对比像素：遍历图片中原始水印有内容的位置（透明通道>0 的区域），根据水印颜色（深色/浅色）判断该位置是否检测到水印。

4、计算相似度：（匹配的像素数/原始水印总像素数）×100%。

七、鲁棒性测试

```
def test_robustness(self, original_image_path,
output_dir="custom_results"):
```

整合前面的功能，执行完整的水印鲁棒性测试流程。

1、创建输出目录，用于保存测试结果。

2、调用 embed_watermark 生成含水印的图片，并保存。

3、对含水印的图片依次应用所有支持的变换，保存变换后的图片。

4、对每个变换后的图片调用 detect_watermark，计算水印相似度。

5、打印并返回所有变换的测试结果。