缩放 PDF 内容以适应打印

我需要打印文件，但发现内容周围的空白区域通常太大，浪费纸张并使文本显得比应有的小。这个脚本通过检测内容区域并将其放大以填充页面，同时保留较小的边距，从而自动缩放 PDF 内容以更好地适应页面。

```
import subprocess
import sys
import os
from PIL import Image
from pdf2image import convert_from_path

MARGIN_PERCENT = 0.005
DPI = 72

def convert_pixels_to_points(pixels, dpi):
    """ 将像素转换为点。 """
    return pixels * 72 / dpi

def get_image_dimensions(image):
    """ 获取图像的像素和点尺寸。 """
    width, height = image.size
    dpi = image.info.get('dpi', (DPI, DPI))
    width_points = convert_pixels_to_points(width, dpi[0])
    height_points = convert_pixels_to_points(height, dpi[1])
    return width, height, width_points, height_points, dpi

def analyze_whitespace(image, width, height):
    """ 分析空白区域以找到内容边界框。 """
    left_margin_px = width
    right_margin_px = 0
    top_margin_px = height
    bottom_margin_px = 0
    found_content = False

    for x in range(width):
        for y in range(height):
            pixel = image.getpixel((x, y))
            if isinstance(pixel, tuple):
                if any(c < 250 for c in pixel):
                    if not found_content:
```

```
                        left_margin_px = x
                        top_margin_px = y
                        found_content = True
                    right_margin_px = max(right_margin_px, x)
                    bottom_margin_px = max(bottom_margin_px, y)
            elif pixel < 250:
                if not found_content:
                    left_margin_px = x
                    top_margin_px = y
                    found_content = True
                right_margin_px = max(right_margin_px, x)
                bottom_margin_px = max(bottom_margin_px, y)

    if not found_content:
        return None, None, None, None

    right_margin_px = width - right_margin_px
    bottom_margin_px = height - bottom_margin_px
    return left_margin_px, right_margin_px, top_margin_px, bottom_margin_px

def calculate_scale_factor(input_pdf):
    """
    检测 PDF 第一页的尺寸，分析空白区域，
    并根据 PDF 内容和目标 A4 尺寸（带边距）计算缩放因子。
    返回缩放因子，如果发生错误则返回 None。
    """
    print(f" 计算缩放因子：{input_pdf}")
    try:
        images = convert_from_path(input_pdf, first_page=1, last_page=1)
        if not images:
            print("  无法将 PDF 转换为图像。")
            return None

        image = images[0]
        width, height, width_points, height_points, dpi = get_image_dimensions(image)

        margins = analyze_whitespace(image, width, height)
        if margins[0] is None:
            print("  无法确定内容边界框。")
            left_margin_points = 0
```

```
        right_margin_points = 0
        top_margin_points = 0
        bottom_margin_points = 0
    else:
        left_margin_px, right_margin_px, top_margin_px, bottom_margin_px = margins
        content_width_px = right_margin_px - left_margin_px
        content_height_px = bottom_margin_px - top_margin_px

        left_margin_points = convert_pixels_to_points(left_margin_px, dpi[0])
        right_margin_points = convert_pixels_to_points(right_margin_px, dpi[0])
        top_margin_points = convert_pixels_to_points(top_margin_px, dpi[1])
        bottom_margin_points = convert_pixels_to_points(bottom_margin_px, dpi[1])

        print(f"  内容框：左 ={left_margin_px}，上 ={top_margin_px}，右 ={right_margin_px}，下 ={bottom_m
        print(f"  内容尺寸（像素）：宽度 ={content_width_px}，高度 ={content_height_px}")
        print(f"  边距（点）：左 ={left_margin_points}，右 ={right_margin_points}，上 ={top_margin_points}

    print(f"  检测到的尺寸：宽度 ={width_points}，高度 ={height_points}")

    width_margin_points = min(left_margin_points, right_margin_points)
    height_margin_points = min(top_margin_points, bottom_margin_points)

    content_width = width_points - width_margin_points * 2
    content_height = height_points - height_margin_points * 2

    target_width = width_points * (1 - 2 * MARGIN_PERCENT)
    target_height = height_points * (1- 2 * MARGIN_PERCENT)

    width_scale = target_width / content_width
    height_scale = target_height / content_height

    print(f"  内容尺寸（点）：宽度 ={content_width}，高度 ={content_height}")

    if content_width <= 0 or content_height <= 0:
        print(" 错误：无法确定内容尺寸。")
        return None

    print(f"  目标尺寸：宽度 ={target_width}，高度 ={target_height}")
    print(f"  计算出的宽度缩放：{width_scale}，高度缩放：{height_scale}")
```

```
        scale_factor = min(width_scale, height_scale)
        print(f"  最终缩放因子：{scale_factor}")

        return scale_factor

    except Exception as e:
        print(f" 获取 PDF 尺寸或计算缩放因子时出错：{e}")
        return None


def scale_pdf(input_pdf, output_pdf, scale_factor):
    """ 使用 pdfjam 缩放 PDF。 """
    print(f" 将 {input_pdf} 缩放至 {output_pdf}，缩放因子：{scale_factor}")
    try:
        subprocess.run(
            [
                "pdfjam",
                "--scale",
                str(scale_factor),
                input_pdf,
                "--outfile",
                output_pdf,
            ],
            check=True,
        )
        print(f" 成功将 {input_pdf} 缩放至 {output_pdf}")
    except subprocess.CalledProcessError as e:
        print(f" 缩放 PDF 时出错：{e}")
    except FileNotFoundError:
        print(" 错误：未找到 pdfjam 命令。请确保已安装并将其添加到系统的 PATH 中。 ")


if __name__ == "__main__":
    if len(sys.argv) != 3:
        print(" 用法：python scale-pdf.py <input_pdf> <output_pdf>")
        sys.exit(1)

    input_pdf = sys.argv[1]
    output_pdf = sys.argv[2]
    print(f" 输入 PDF：{input_pdf}，输出 PDF：{output_pdf}")
```

```
if not os.path.exists(input_pdf):
    print(f" 错误：未找到输入 PDF 文件：{input_pdf}")
    sys.exit(1)

scale_factor = calculate_scale_factor(input_pdf)
if scale_factor is None:
    sys.exit(1)

scale_pdf(input_pdf, output_pdf, scale_factor)
```