Scale PDF Content for Printing

I need to print papers, and I've noticed that the whitespace around the content is often too large, wasting paper and making the text appear smaller than it should. This script helps to automatically scale the PDF content to better fit the page by detecting the content area and scaling it up to fill the page, while respecting a small margin.

```
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):
    """Converts pixels to points."""
    return pixels * 72 / dpi

def get_image_dimensions(image):
    """Gets image dimensions in pixels and points."""
    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):
    """Analyzes whitespace to find content bounding box."""
    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):
    """
    Detects the dimensions of the first page of a PDF, analyzes whitespace,
    and calculates the scale factor based on the PDF content and target A4 dimensions with margins.
    Returns the scale factor or None if an error occurs.
    """
    print(f"Calculating scale factor for: {input_pdf}")
    try:
        images = convert_from_path(input_pdf, first_page=1, last_page=1)
        if not images:
            print("  Could not convert PDF to image.")
            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("  Could not determine content bounding box.")
            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"  Content box: left={left_margin_px}, upper={top_margin_px}, right={right_margin_px}, lowe
            print(f"  Content dimensions (pixels): width={content_width_px}, height={content_height_px}")
            print(f"  Margins (points): left={left_margin_points}, right={right_margin_points}, top={top_margi

        print(f"  Detected dimensions: width={width_points}, height={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"  Content dimensions (points): width={content_width}, height={content_height}")

        if content_width <= 0 or content_height <= 0:
            print("Error: Could not determine content dimensions.")
            return None

        print(f"  Target dimensions: width={target_width}, height={target_height}")
        print(f"  Calculated width scale: {width_scale}, height scale: {height_scale}")
```

```python
        scale_factor = min(width_scale, height_scale)
        print(f"  Final scale factor: {scale_factor}")

        return scale_factor

    except Exception as e:
        print(f"Error getting PDF dimensions or calculating scale factor: {e}")
        return None


def scale_pdf(input_pdf, output_pdf, scale_factor):
    """Scales a PDF using pdfjam."""
    print(f"Scaling {input_pdf} to {output_pdf} with scale factor: {scale_factor}")
    try:
        subprocess.run(
            [
                "pdfjam",
                "--scale",
                str(scale_factor),
                input_pdf,
                "--outfile",
                output_pdf,
            ],
            check=True,
        )
        print(f"Successfully scaled {input_pdf} to {output_pdf}")
    except subprocess.CalledProcessError as e:
        print(f"Error scaling PDF: {e}")
    except FileNotFoundError:
        print("Error: pdfjam command not found. Please ensure it is installed and in your system's PATH.")


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

    input_pdf = sys.argv[1]
    output_pdf = sys.argv[2]
```

```python
print(f"Input PDF: {input_pdf}, Output PDF: {output_pdf}")

if not os.path.exists(input_pdf):
    print(f"Error: Input PDF file not found: {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)
```