Py_to_PDF

May 8, 2025

[ ]:
```
import time
import math
import matplotlib
matplotlib.use("TkAgg")  # Use TkAgg backend instead of Qt
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import numpy as np
from PIL import Image

# Try importing Picamera2 for Raspberry Pi capture
try:
    from picamera2 import Picamera2
except ImportError:
    Picamera2 = None

def capture_image():
    """
    Captures a single image using the Raspberry Pi camera (Picamera2).
    Returns a PIL Image object.
    """
    if Picamera2 is None:
        print("Picamera2 is not available. Make sure you are running on a␣
 ↪Raspberry Pi with Picamera2 installed.")
        return None

    picam2 = Picamera2()
    # Configure for still capture with desired resolution
    config = picam2.create_still_configuration(main={"size": (640, 480)})
    picam2.configure(config)
    picam2.start()

    # Allow the camera time to adjust (exposure, focus, etc.)
    time.sleep(2)
    frame = picam2.capture_array()
    picam2.stop()

    return Image.fromarray(frame)
```

```python
def pick_line_and_get_rotation_angle(pil_img):
    """
    Opens the provided PIL image in a popup window for interactive clicks.
    Lets the user click 2 points to define a line and computes the rotation angle (in degrees)
    required to rotate the image so that the line becomes horizontal.
    No additional popup window is shown for displaying the selected line.

    Returns the rotation angle (in degrees) to apply.
    """
    img_arr = np.array(pil_img)
    plt.figure()
    plt.imshow(img_arr)
    plt.title("Click 2 points to define a line for rotation alignment")

    # Capture exactly 2 points interactively
    pts = plt.ginput(n=2, timeout=0)
    plt.close()

    if len(pts) < 2:
        print("Not enough points selected for line alignment.")
        return None

    (x1, y1), (x2, y2) = pts
    # Compute the angle (in radians) of the line relative to the horizontal axis
    angle_rad = math.atan2(y2 - y1, x2 - x1)
    # Convert the angle to degrees
    angle_deg = math.degrees(angle_rad)
    # Apply the computed angle directly for rotation.
    # (Remove the negative sign so that the image rotates by the computed angle)
    rotation_angle = angle_deg

    print(f"Selected points: (x1={x1:.2f}, y1={y1:.2f}), (x2={x2:.2f}, y2={y2:.2f})")
    print(f"Computed line angle: {angle_deg:.2f}°; Applying rotation angle: {rotation_angle:.2f}°")

    return rotation_angle

def pick_points_and_show_crop_box_from_image(pil_img, n=4):
    """
    Opens the provided PIL image in a popup window for interactive clicks.
    Lets the user click n points to define a cropping rectangle, then displays the image
    with a yellow rectangle showing the crop area.
```

```
    Returns the crop box as (left, top, right, bottom).
    """
    img_arr = np.array(pil_img)
    plt.figure()
    plt.imshow(img_arr)
    plt.title(f"Click {n} points to define the crop rectangle")

    # Capture clicks interactively
    pts = plt.ginput(n=n, timeout=0)
    plt.close()

    if len(pts) < n:
        print(f"Not enough points selected. Clicked only {len(pts)} point(s).")
        return None

    xs = [p[0] for p in pts]
    ys = [p[1] for p in pts]
    left, right = min(xs), max(xs)
    top, bottom = min(ys), max(ys)

    print("Clicked crop points:")
    for i, (xv, yv) in enumerate(pts, start=1):
        print(f"  Point #{i}: (x={xv:.2f}, y={yv:.2f})")
    print(f"Computed Crop Box => left={left:.2f}, top={top:.2f}, right={right:.
↪2f}, bottom={bottom:.2f}")

    # Display the image with the crop rectangle
    fig, ax = plt.subplots(figsize=(8, 6))
    ax.imshow(img_arr)

    width = right - left
    height = bottom - top

    rect = Rectangle((left, top), width, height,
                     edgecolor="yellow", facecolor="none", linewidth=2)
    ax.add_patch(rect)
    ax.set_title("Image with Crop Box (yellow)")
    ax.axis("off")
    plt.show()

    return (left, top, right, bottom)

if __name__ == "__main__":
    # Step 1: Capture an image using the Pi camera
    captured_img = capture_image()
    if captured_img is None:
        print("Image capture failed. Check your camera configuration.")
```

```
    else:
        # Step 2: Get the rotation angle from a user-clicked line (only one␣
↪popup)
        rotation_angle = pick_line_and_get_rotation_angle(captured_img)
        if rotation_angle is not None:
            # Rotate the image so that the line becomes horizontal (using the␣
↪computed angle)
            rotated_img = captured_img.rotate(rotation_angle, expand=True)
            print(f"Rotated image by {rotation_angle:.2f}°")

            # Optionally, save the rotated image for verification
            rotated_img.save("rotated_image.jpg")

            # Step 3: Use the rotated image for interactive cropping
            crop_box = pick_points_and_show_crop_box_from_image(rotated_img,␣
↪n=4)
            if crop_box:
                print(f"Final crop box: {crop_box}")
```