Function Documentation: `process_series`

1 Overview

The `process_series` function processes a series of images, applying transformations and saving the results. It uses the following parameters:

- `series_prefix`: Prefix for output file names.
- `series`: Tuple containing series information including file paths, mask, dish coordinates, etc.
- `min_radius`: Minimum radius for Petri dish extraction.
- `min_count`: Minimum count for filtering files.

2 Function Definition

2.1 Function Header

```
def process_series(series_prefix, series, min_radius, min_count):
```

2.2 Unpack Series Information

```
date, scanner, experiment, files, mask, dish_coords, dish_angles, first_frame, last_frame
↪  = series
for i, (x, y, r) in enumerate(dish_coords):
↪  print(f'output/{series_prefix}_{i+1:02}_d03.jpg')
```

Explanation: Unpack the tuple `series` into its components and print file paths for each dish coordinate.

2.3 Check Dish Count

```
if len(dish_coords) < 6:
    print(f'WARNING: PREPROCESSED/{series_prefix}_%d_%03d has {len(dish_coords)} dishes,
    ↪  expected 6')
```

Explanation: Issue a warning if the number of detected dishes is less than 6.

2.4 Get the First Good Image

```
last_frame = {}
for filepath in files:
    try:
        frame = cv2.imread(filepath)
        masked_frame = cv2.bitwise_and(frame, mask)
        for i, (x, y, r) in enumerate(dish_coords):
            x, y, r = max(x, min_radius), max(y, min_radius), min(min_radius, min(x, y,
            ↪  masked_frame.shape[1] - x, masked_frame.shape[0] - y))
            dish = masked_frame[y - min_radius:y + min_radius, x - min_radius:x +
            ↪  min_radius]
            dish = apply_curve_transformation(dish, piecewise_linear_transformation, 0.0,
            ↪  0.85)
            last_frame[i] = dish
    except:
        continue
```

Explanation: Read each image, apply the mask, extract and transform the Petri dish regions, and store them in `last_frame`.

2.5 Process Each Image

```
this_frame_index = 1
frame_index = 0
for filepath in files:
    frame = cv2.imread(filepath)
    frame_index = int(filepath.split('_')[-1].split('.')[0])
    if frame is None:
        continue
    while this_frame_index < frame_index:
        for i, (x, y, r) in enumerate(dish_coords):
            this_last_frame = last_frame[i].copy()
            cv2.putText(this_last_frame, f'f{this_frame_index:03}', (105, 105),
            ↪  cv2.FONT_HERSHEY_SIMPLEX, 3, (255, 255, 255), 2)
            cv2.circle(this_last_frame, (50, 50), 40, (50, 50, 250), -1)

            ↪  cv2.imwrite(f'PREPROCESSED/{series_prefix}_{i+1:02}_{this_frame_index:03}.jpg',
            ↪  this_last_frame)
        this_frame_index += 1
```

Explanation: For each file, create images for missing frames by using the last good frame data.

2.6 Process Current Frame

```
if dish_coords is not None and mask is not None:
    masked_frame = cv2.bitwise_and(frame, mask)
    for i, (x, y, r) in enumerate(dish_coords):
        x, y, r = max(x, min_radius), max(y, min_radius), min(min_radius, min(x, y,
        ↪  masked_frame.shape[1] - x, masked_frame.shape[0] - y))
        dish = masked_frame[y - min_radius:y + min_radius, x - min_radius:x + min_radius]
        if dish_angles[i] != 0:
            rotation_matrix = cv2.getRotationMatrix2D((int(min_radius),int(min_radius)),
            ↪  dish_angles[i]*180/np.pi, 1)
            dish = cv2.warpAffine(dish, rotation_matrix, (2*int(min_radius),
            ↪  2*int(min_radius)))
        if dish.size > 0 and dish.shape[0] > 0 and dish.shape[1] > 0:
            dish = apply_curve_transformation(dish, piecewise_linear_transformation, 0.0,
            ↪  0.85)
            last_frame[i] = dish.copy()
            cv2.circle(dish, (50, 50), 40, (50, 250, 50), -1)
            cv2.putText(dish, f'f{frame_index:03}', (105, 105), cv2.FONT_HERSHEY_SIMPLEX,
            ↪  3, (255, 255, 255), 2)
            cv2.imwrite(f'PREPROCESSED/{series_prefix}_{i+1:02}_{frame_index:03}.jpg',
            ↪  dish)
    this_frame_index += 1
```

Explanation: For the current frame, apply the mask, extract and transform the Petri dish regions, rotate if necessary, and save the processed image.

2.7 Main Execution Block

```
if __name__ == '__main__':
    args = docopt(__doc__)
    patterns = args['<patterns>']

    ensure_directory_exists('PREPROCESSED')

    min_radius = 0
    min_count = 0

    series_dict = {}
    for date_dir in sorted(next(os.walk('.'))[1]):
        if 'git' not in date_dir and 'PROCESSED' not in date_dir:
            print(date_dir)
            for scanner_dir in sorted(next(os.walk(date_dir))[1]):
                pattern = date_dir + '/' + scanner_dir + '/*'
                files = sorted(glob.glob(pattern))
                if len(files) > 0:
                    experiment_number = files[0].split('.')[0].split('/')[-1]
                    key = date_dir + '_' + scanner_dir + '_' + experiment_number
                    if sum([p in key for p in patterns]):
                        first_frame = int(files[0].split('_')[-1].split('.')[0])
                        last_frame = int(files[-1].split('_')[-1].split('.')[0])
```

```
                        mask, dish_circles, rot_angles = extract_radius_mask(files[0],
                        ↪  save=f'PREPROCESSED/preprocess_{key}.jpg')
                        series_dict[key] = [
                                date_dir,
                                scanner_dir,
                                experiment_number,
                                files,
                                mask,
                                dish_circles,
                                rot_angles,
                                first_frame,
                                last_frame
                        ]
                        if max(dish_circles[:,2]) > min_radius:
                            min_radius = max(dish_circles[:,2])
                        if len(files) > min_count:
                            min_count = len(files)

    for key, series in series_dict.items():
        process_series(key, series, min_radius, min_count)
```

Explanation: The script initializes parameters, collects series data from directories, and processes each series using `process_series`.