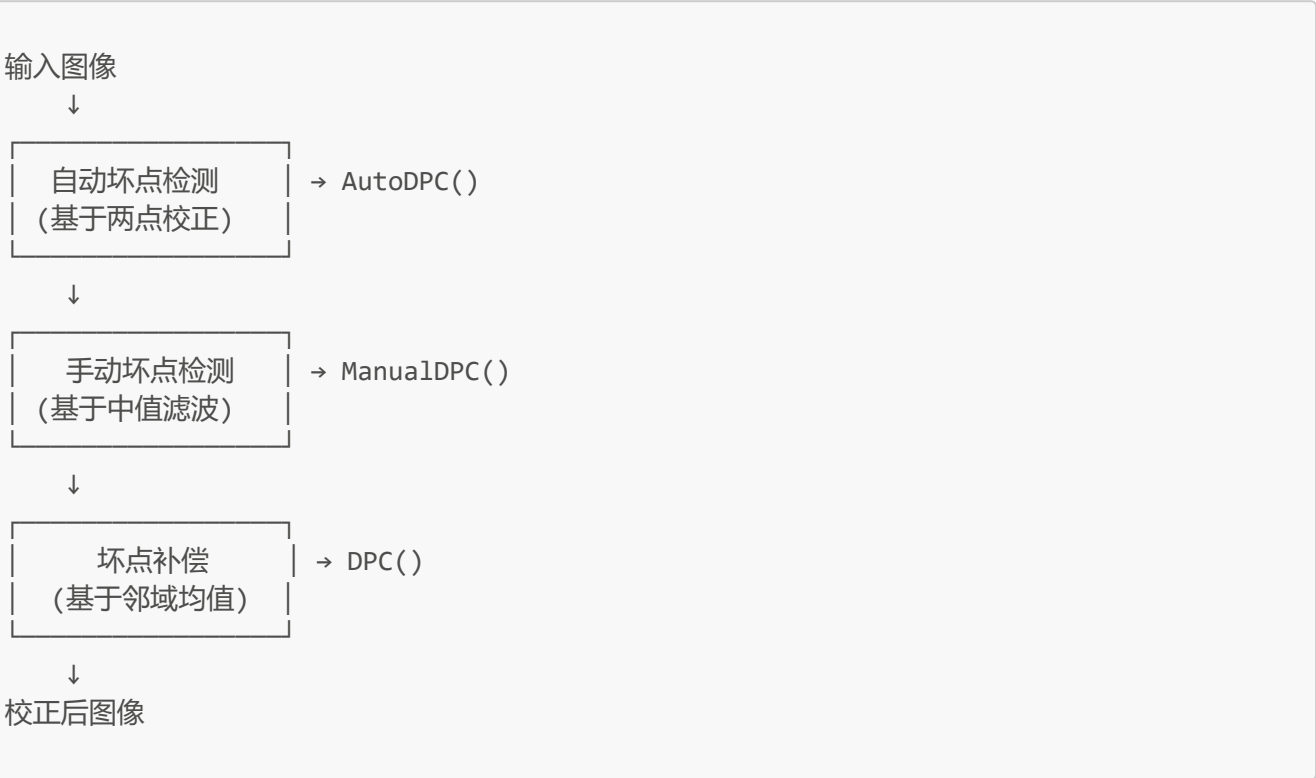
红外图像去坏点算法详细技术文档

1. 算法概述

本算法实现了一套完整的红外图像坏点检测与校正系统，包含自动坏点检测、手动坏点检测和坏点校正三个核心模块。算法能够有效处理坏点（Dead Pixel）和盲元（Stuck Pixel），并提供高质量的图像校正结果。

2. 算法架构



3. 核心算法详解

3.1 自动坏点检测算法 (AutoDPC)

3.1.1 实现细节

步骤1：计算响应差值

```
Dif = double(hot_uniform) - double(cold_uniform);
Dif_temp = double(hot_temp) - double(cold_temp);
```

步骤2：计算归一化系数矩阵

```
k = Dif ./ Dif_temp;
```

其中k值反映了像素的温度响应特性。

步骤3：坏点检测

```
if Dif(i, j) == 0
    DeadPixel_Lis = [DeadPixel_Lis; [i, j]];
end
```

当像素在两个温度下的响应差为0时，判定为坏点。

步骤4：盲元检测

- 使用3×3窗口计算当前像素k值与邻域k值的中位数差异
- 排除已检测到的坏点和盲元
- 当差异超过阈值时判定为盲元

```
k_med = median(k_vld_vals);
if abs(k_this - k_med) > thres
    StuckPixel_Lis = [StuckPixel_Lis; [i, j]];
end
```

3.1.2 优化

- **边界处理**：统一采用复制 padding 进行边界扩展
- **有效像素筛选**：动态更新已检测的坏点，避免已有坏点对后续检测的影响
- ☐ 后续是否考虑把坏点和盲元自动检测合并？（理论上可行）

3.1.3 实验结果

- 1x2，2x1，1x3，3x1，1+2x1，2x2 都能解决
- 3x3 能识别出坏点，但是中心点校正出错，原因在于周围一圈都是坏点，均值替换错误

> **按均值校正原理，需要保证一个坏点周围一圈至少有一个正常点**

3.2 手动坏点检测算法 (ManualDPC)

3.2.1 实现细节

步骤1：邻域提取

```
row_start = max(1, i - floor(MedWindow_size / 2));
row_end = min(h, i + floor(MedWindow_size / 2));
col_start = max(1, j - floor(MedWindow_size / 2));
col_end = min(w, j + floor(MedWindow_size / 2));
window = image_input(row_start:row_end, col_start:col_end);
```

步骤2：坐标系转换 将全局坐标系中的已知坏点转换为窗口局部坐标系：

```
AllDP_Lis_obj = [AllDP_Lis(k, 1)-row_start+1, AllDP_Lis(k, 2)-col_start+1];
```

步骤3：窗口内坏点检测
调用MedForDPC函数在5×5窗口内进行中值滤波检测。

步骤4：动态更新机制

```
AllDP_Lis = [AllDP_Lis; global_coord];
AllDP_Lis = sortrows(AllDP_Lis);
```

每检测到新坏点立即更新全局坏点列表并排序。

3.2.2 关键特性

- **渐进式检测**：基于已有检测结果进行增量检测
- **坐标系管理**：正确处理全局与局部坐标系的转换
- **实时更新**：保证后续检测能利用最新的坏点信息

3.2.3 实验结果

- 单坏点 5x5 内坏点判断正确
- 当选中位置的边角出现连着的坏点，边角无法判断为坏点

  边角出现连着的坏点的情况，中心点为手动检测选中的坐标：
  □□□□□
  □□□□□
  □□□□□
  □□□□□
  □□□■■

> **复制 padding 以后， 边角所在邻域有五个坏点，但是再扩大邻域 可能太吃 fpga 资源**

3.3 中值滤波坏点检测 (MedForDPC)

3.3.1 算法原理

在指定窗口内使用3×3中值滤波进行坏点检测，基于像素值与其邻域中值的差异判断。

核心思想：

- 正常像素值应接近其邻域的中值
- 坏点像素值与邻域中值存在显著差异
- 通过阈值比较进行判定

3.3.2 实现细节

步骤1：窗口边界扩展

```
window_padded = padarray(window, [Padded_size, Padded_size], 'replicate');
```

步骤2：邻域像素收集

```
for ii = i-Padded_size:i+Padded_size
    for jj = j-Padded_size:j+Padded_size
        % 排除中心像素和已知坏点
        if ((ii == i && jj == j)) ||
           ismember([ii-Padded_size, jj-Padded_size], DeadPixel_Lis, 'rows')
            continue;
        end
        MedPixel_Vals = [MedPixel_Vals; window_padded(ii, jj)];
    end
end
```

步骤3：中值计算与判定

```
MedPixel_Val = median(MedPixel_Vals);
if abs(window_padded(i, j) - MedPixel_Val) > thres_Med
    % 检测到坏点
    DeadPixel_Lis = [DeadPixel_Lis; [i-Padded_size, j-Padded_size]];
    ManualPixel_Lis = [ManualPixel_Lis; [i-Padded_size, j-Padded_size]];
end
```

3.3.3 动态更新机制

- **实时坏点列表更新**：每检测到新坏点立即添加到`DeadPixel_Lis`
- **影响后续检测**：新检测的坏点会被排除在后续邻域计算之外
- **坐标一致性**：维护窗口局部坐标系的一致性

3.4 坏点校正算法 (DPC)

3.4.1 算法原理

使用邻域均值插值方法校正检测到的坏点，基于周围正常像素的加权平均。

校正策略：

- 3×3邻域均值插值
- 排除邻域内的其他坏点
- 确保使用有效像素进行计算
- *要求3x3区域必须有有效像素点*

3.4.2 实现细节

步骤1：坏点判定

```
if ismember([i, j], AllDP_Lis, 'rows')
    % 当前像素是坏点，需要校正
end
```

步骤2：有效邻域统计

```
for ii = row_start:row_end
    for jj = col_start:col_end
        if ismember([ii, jj], AllDP_Lis, 'rows')
            continue; % 跳过邻域内的坏点
        end
        valid_cnt = valid_cnt + 1;
        valid_sum = valid_sum + double(padded_image_in(ii+pad_size, jj+pad_size));
    end
end
```

步骤3：均值计算与替换

```
mean_val = valid_sum / valid_cnt;
image_dpc(i, j) = uint16(mean_val);
```

4. 参数配置说明

4.1 关键参数

参数名	默认值	作用	调优建议（AI）
`thres`	100	自动检测阈值	根据图像动态范围调整
`thres_Med`	30	手动校正中值滤波阈值	根据噪声水平调整
`MeanWindow_size`	3	校正窗口大小	可选3×3或5×5
`MedWindow_size`	5	手动检测窗口	根据坏点聚集程度调整

4.2 参数选择准则

阈值设定：

- `thres`：通常设为图像动态范围的5-10%
- `thres_Med`：建议为噪声标准差的2-3倍

窗口大小：

- 检测窗口：3×3适合孤立坏点，5×5适合聚集坏点

- 校正窗口：3×3提供较好的细节保持

5. 算法性能分析

5.1 时间复杂度

- **AutoDPC**: O(n^2) - 全图扫描
- **ManualDPC**: O(m × k^2) - m个手动点，k×k窗口
- **MedForDPC**: O(w^2 × 9) - w×w窗口内3×3检测
- **DPC**: O(n^2) - 全图校正

5.2 空间复杂度

- **主要存储**: O(n) - 坏点列表
- **临时存储**: O(k^2) - 处理窗口

5.3 检测准确性

优势：

- 双模式检测提高覆盖率
- 动态更新提高精度
- 多阈值判定减少误检

限制：

- 依赖于校正图像质量
- 聚集坏点可能影响邻域统计
- 边界处理可能引入轻微误差