采集端通信协议规范

采集端通信协议规范

版本: 2.1.3

日期: 2026-02-07

基于: Intro.md (PUB-SUB 模式), ZMQ 协议帧设计规范 .md (v1.5.0)

---

1. 协议概述

1.1 设计背景

随着系统架构演进，新的下位硬件将承担相机采集和预处理功能，替代原有 PipelineV1 中的采集和预处理模块。本协议定义了 ConfigServer 与采集端之间的通信规范，确保系统平滑迁移。

版本 2.1.0 变更:

- 修改了温度数据上传格式
- 增加了一维数据的标识方法
- 将 float 格式统一为 int 格式

版本 2.1.2 变更:

- 增加了发送配置的定义

版本 2.1.3 变更:

- 增加了描述插图
- 修改了部分表述

1.2 设计原则

1. **简单性**: 采用 PUB-SUB 单向通信模式，降低实现复杂度
2. **高效性**: 针对温度矩阵传输优化，支持实时数据流
3. **可扩展性**: 支持参数动态配置和功能扩展

4. **可靠性**: 完善的错误处理和状态监控机制
5. **兼容性**: 完全兼容 Intro.md 中的下位机协议

1.3 协议适用范围

- ConfigServer 向采集端下发配置参数
- 采集端向 ConfigServer 返回预处理数据
- 采集端状态监控与健康检查
- 实时图像流传输
- 多设备管理（基于设备 ID）

---

2. 系统架构与数据流

2.1 系统架构

Plain Text

```
graph TD
    A[上位机/UI] --> B[ConfigServer]
    B --> C[采集端硬件]
    C --> D[相机采集]
    D --> E[预处理模块]
    E --> F[温度矩阵输出]
    F --> B
    B --> G[Pipeline预测器]

    subgraph "PUB-SUB通信"
        H[ConfigServer PUB] --> I[发布命令/配置]
        J[ConfigServer SUB] --> K[接收数据/状态]
        L[采集端 SUB] --> M[接收命令/配置]
        N[采集端 PUB] --> O[发布数据/状态]
    end

    style C fill:#e1f5e1
    style G fill:#fff3e0
    style H fill:#e3f2fd
    style J fill:#f3e5f5
```

2.2 数据流说明

2.2.1 命令下发流 (ConfigServer → 采集端)

- **配置参数更新**: 通过主题 `{deviceId}/setting` 下发
- **控制命令**: 通过主题 `{deviceId}/command` 下发
- **文件传输**: 通过主题 `{deviceId}/file` 下发
- **设备注册响应**: 通过主题 `{deviceId}/ready` 下发

2.2.2 数据返回流 (采集端 → ConfigServer)

- **设备注册**: 通过主题 `register` 上报
- **命令结果**: 通过主题 `result` 上报
- **状态信息**: 通过主题 `status` 上报
- **处理完成回执**: 通过主题 `changed` 上报
- **帧数据**: 通过主题 `acquisition.*` 上报

2.2.3 数据转发流 (ConfigServer → Pipeline 预测器)

- 预处理后的温度矩阵
- 触发事件信息

2.3 端口配置

方向	ConfigServer 端口	采集端端口	用途	主题示例
下发命令	5511 (PUB)	5512 (SUB)	命令下发	`device001/setting`
接收数据	5511 (SUB)	5512 (PUB)	数据上报	`result`, `register`

端口说明:

- **5511 端口**: ConfigServer 作为 PUB 发布者，采集端作为 SUB 订阅者
- **5512 端口**: ConfigServer 作为 SUB 订阅者，采集端作为 PUB 发布者

---

3. PUB-SUB 协议设计

3.1 帧结构

采用标准的 PUB-SUB 帧结构，基于 ZMTP 3.0 协议：

Plain Text

```
┌──────────────────────────────────────────────────────────────────────┐
│                        ZMQ Multipart Message (PUB-SUB)               │
├──────────────────┬──────────────────┬────────────────────────────────┤
│   Frame 0        │   Frame 1        │          Frame 2               │
│   (Topic)        │   (Data)         │        (可选扩展)              │
├──────────────────┼──────────────────┼────────────────────────────────┤
│   主题字符串     │  序列化数据      │          扩展数据              │
└──────────────────┴──────────────────┴────────────────────────────────┘
```

3.2 ZMTP 握手流程

Plain Text

```
客户端（采集端）                服务端（ConfigServer）
      │                                  │
      │  1. GREETING (10 bytes)          │
      │─────────────────────────────────>│
      │                                  │
      │                          2. GREETING (10 bytes)
      │<─────────────────────────────────│
      │                                  │
      │  3. READY (Command + Metadata)   │
      │─────────────────────────────────>│
      │                                  │
      │                          4. READY (Command + Metadata)
      │<─────────────────────────────────│
      │                                  │
      │            握手完成，开始消息传输           │
```

GREETING 格式（64 字节）：

- 签名（10 字节）：`0xFF 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x7F`
- 版本（2 字节）：`{3, 0}`（ZMTP 3.0）
- 机制（20 字节）："NULL" + 16 个 0
- as_server（1 字节）：0x00（客户端）或 0x01（服务端）
- 填充（31 字节）：0

3.3 主题命名规范

3.3.1 设备专属主题

- `{deviceId}/setting` - 配置参数下发
- `{deviceId}/command` - 控制命令下发
- `{deviceId}/file` - 文件传输
- `{deviceId}/ready` - 设备就绪信号
- `{deviceId}/response/success` - 成功响应
- `{deviceId}/response/error` - 错误响应

3.3.2 通用主题

- `register` - 设备注册
- `result` - 检测结果
- `status` - 状态信息
- `changed` - 处理完成回执
- `acquisition.frame.triggered` - 触发帧数据
- `acquisition.frame.masked` - 蒙版帧数据
- `acquisition.status` - 采集状态

3.3.3 主题匹配规则

- 前缀匹配：订阅 `device001/` 将接收所有 `device001/` 开头的消息
- 精确匹配：订阅 `register` 只接收 `register` 消息

---

4. 参数下发协议

4.1 命令帧结构

所有命令通过 PUB-SUB 模式下发：

```
Plain Text
```

```
1 Frame 0: Topic (主题字符串，如 "device001/setting")
2 Frame 1: Payload (参数数据，JSON格式)
```

4.2 命令定义

4.2.1 配置更新命令 (`setting` 主题)

用于向采集端下发完整的配置参数。根据一维 / 二维的任务需求 , 传递变长参数

当使用一维任务时 ,Camera 配置项的 **DeviceId** 一定是 -1, 代表不使用相机

Topic 格式: `{deviceId}/setting`

Payload 格式 (JSON 二维示例):

JSON


```
{
  "RequestId": "req_config_001",
  "PipelineId": "line1",
  "timestamp": "2026-02-03T08:30:00Z",
  "value": 10,
  "config": {
    "Version": 2,
    "Command": "ConfigureAcquisition",
    "RequestId": "req_config_001",
    "PipelineId": "line1",
    "Parameters": {
      "Basic": {
        "PipelineId": "line1",
        "PipelineType": "TemperatureDetection",
        "Mode": "Detect",
        "ConfigTag": "stand",
        "StrictnessLevel": 1,
        "IsCustomMode": false
      },
      "Camera": {
        "DeviceId": 1,
        "Width": 384,
        "Height": 288,
        "VideoMode": "TMP",
        "Fps": 30,
        "Exposure": 10000,
        "AutoExposure": false,
        "CaptureTimeoutMs": 100
      },
      "Mask": {
        "Enabled": true,
        "Threshold": 30.0,
        "Width": 185,
        "Height": 70,
        "Angle": 0.0,
        "TargetWidth": 185,
        "TargetHeight": 70
      },
      "RegionOfInterest": {
```

```
        "X": 0,
        "Y": 0,
        "Width": 160,
        "Height": 40
      },
      "Trigger": {
        "TriggerGpioLine": 1,
        "Mode": "External",
        "DelayMs": 0,
        "BurstCount": 1,
        "InternalIntervalMs": 100,
        "TemperatureThreshold": 45.0,
        "DebounceIntervalMs": 0,
        "TriggerRoi": {"X":0,"Y":0,"Width":0,"Height":0},
        "TriggerCondition": "Average"
      },
      "Output": {
        "OutputGpioLine": 2,
        "AlarmGpioLine": 3,
        "AlarmHoldMs": 1000,
        "AlarmDebounceMs": 200
      },
      "Storage": {
        "StoreNgImagesOnly": true
      },
      "Training": {
        "Enabled": false,
        "SampleThreshold": 20,
        "MaxCachedSamples": 200,
        "AutoTrigger": true
      },
      "System": {
        "ProcessingTimeoutMs": 100,
        "MaxProcessingQueueSize": 10,
        "EnableDataForwarding": true,
        "CommunicationMode": "PUB_SUB"
      }
    }
  }
}
```

Payload 格式 (JSON 一维示例):

JSON

```json
{
  "RequestId": "req_config_001",
  "PipelineId": "line1",
  "timestamp": "2026-02-03T08:30:00Z",
  "value": 10,
  "config": {
    "Version": 2,
    "Command": "ConfigureAcquisition",
    "RequestId": "req_config_001",
    "PipelineId": "line1",
    "Parameters": {
      "Basic": {
        "PipelineId": "line1",
        "PipelineType": "TemperatureDetection",
        "Mode": "Detect",
        "ConfigTag": "stand",
        "StrictnessLevel": 1,
        "IsCustomMode": false
      },
      "Camera": {
        "DeviceId": -1
      },
      "Settings": {
        "mode": "RUN",
        "high_timer_limit":5,
        "timer_c_limit":5,
        "buffer_size": 5,
        "trigger_temp_limit": 5,
        "left_window_range": 5,
        "start_points_to_remove": 5,
        "reference_length": 5,
        "ng_count_limit":5,
        "trigger":0
      }
    }
  }
}
```

4.2.2 部分参数更新命令

通过 `{deviceId}/setting` 主题下发，使用简化格式：

Payload 格式：

JSON

```
{
  "RequestId": "req_update_001",
  "PipelineId": "line1",
  "timestamp": "2026-02-03T08:31:00Z",
  "command": "UpdateParameters",
  "parameters": {
    "Mask.Threshold": 35.0,
    "Trigger.DelayMs": 50,
    "Camera.Fps": 25
  }
}
```

4.2.3 控制命令 (`command` 主题)

通过 `{deviceId}/command` 主题下发控制命令：

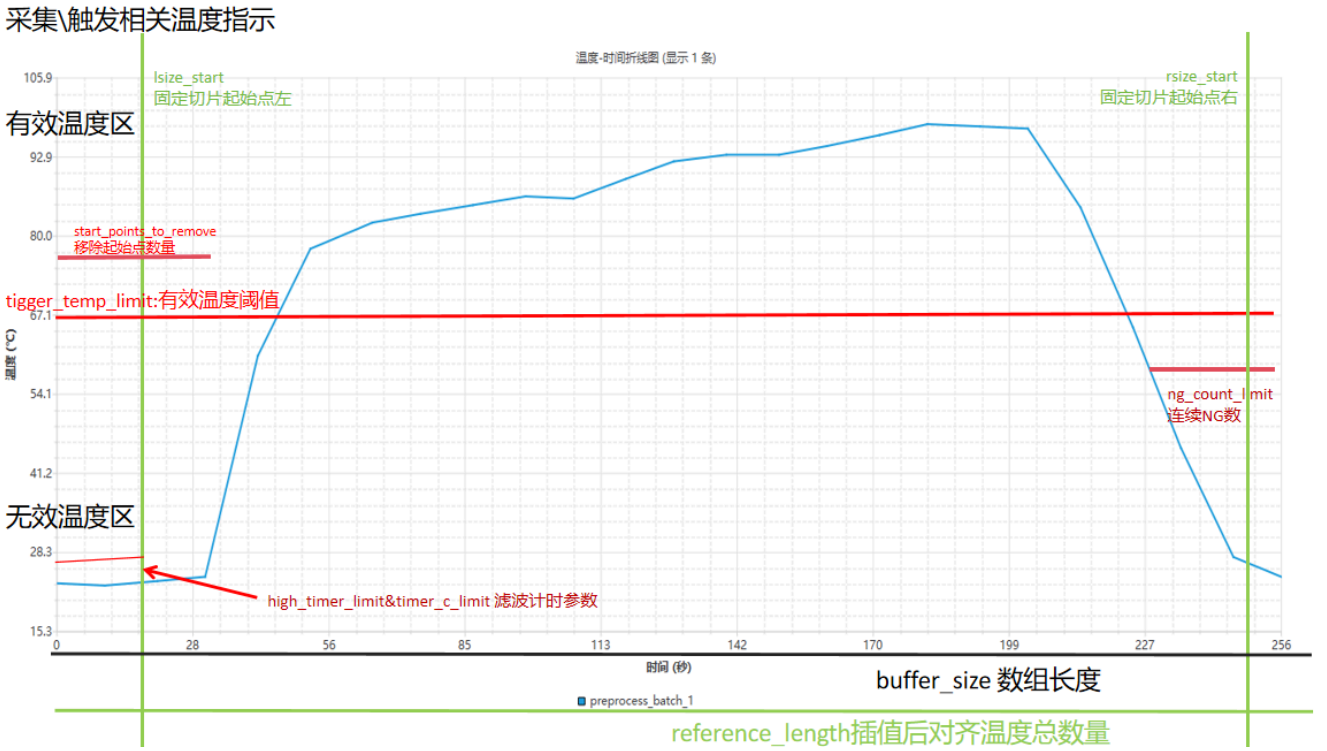
命令名称	说明	参数	Payload 示例
`StartAcquisition`	启动采集	`{"mode": "Detect"}`	`{"command": "StartAcquisition", "PipelineId": "line1", "mode": "Detect"}`
`StopAcquisition`	停止采集	无	`{"command": "StopAcquisition", "PipelineId": "line1"}`
`PauseAcquisition`	暂停采集	无	`{"command": "PauseAcquisition", "PipelineId": "line1"}`
`ResumeAcquisition`	恢复采集	无	`{"command": "ResumeAcquisition", "PipelineId": "line1"}`
`GetStatus`	获取状态	无	`{"command": "GetStatus", "PipelineId": "line1"}`
`GetConfiguration`	获取配置	无	`{"command": "GetConfiguration", "PipelineId": "line1"}`
`SwitchPipelineMode`	切换配方模式	`{"mode": "stand"}`	`{"command": "SwitchPipelineMode", "PipelineId": "line1", "mode": "stand"}`

`ResetStats`	重置统计数据	无	`{"command": "ResetStats", "PipelineId": "line1"}`
`GetScreenshot`	请求实时截图	无	`{"command": "GetScreenshot", "PipelineId": "line1"}`

4.3 参数详细说明

一维专用参数说明 (Settings):

参数名	类型	说明
mode	string	指示工作状态：STOP/RUN/NOP
buffer_size	int	单包采集最长时间(ms) = X/10
trigger_temp_limit	int	触发温度下限
start_points_to_remove	int	起始移除点数，过滤触发后前 N 点
reference_length	int	预处理对齐长度
high_timer_limit	int	高位滤波计时参数
timer_c_limit	int	滤波计时上限参数
ng_count_limit	int	连续 NG 数 - 停止采集触发
lsize_start	int	向右切片起点
rsize_start	int	向左切片起点
trigger	int	触发方式(0- 外部 ,1- 内部)



一维触发方式说明

外部触发

mode 为 RUN 时 , 始终准备采集 , 当 mode 为 STOP 时 , 任何时候不进行采集

DI 收到电平变化信号(低 -> 高)

触发中断 / 定时滤波(high_timer_limit 代表高电平持续时间 ms)

满足滤波时间开始采集

收集温度点 , 当采集到连续三个(常量)高于触发温度 trigger_temp_limit 时

认为正在采集 , 将数据放入数组

终止采集有两种方式 :

1.buffer_size 被塞满 ,

2. 连续采集 ng_count_limit 个低于 trigger_temp_limit 的温度点

切片数组 , 传递[lsize_start,-rsize_start]

数组打包后进入消息发送队列

内部触发

mode 为 RUN 时 , 始终准备采集 , 当 mode 为 STOP 时 , 任何时候不进行采集

采集流程始终进行 , 内部维护数组状态锁 , 长度为 3 的循环缓冲区

收集温度点 , 当采集到连续三个(常量)高于触发温度 trigger_temp_limit 时

认为正在采集 , 将循环缓冲区写入数组前三位 , 从第四位起继续输入

终止采集有两种方式 :

1.buffer_size 被塞满 ,

2. 连续采集 ng_count_limit 个低于 trigger_temp_limit 的温度点

切片数组 , 传递[lsize_start,-rsize_start]

数组打包后进入消息发送队列

二维专用参数说明 :

基本参数 (Basic)

参数名	类型	默认值	说明
PipelineId	string	"line1"	管线标识
PipelineType	string	"TemperatureDetection"	管线类型
Mode	string	"Detect"	运行模式 : Detect/Train
ConfigTag	string	"stand"	配置标签 : loose/stand/strict/ultra/custom
StrictnessLevel	int	1	严格等级 : 0-3
IsCustomMode	bool	false	是否为自定义模式

相机参数 (Camera)

参数名	类型	默认值	说明
DeviceId	int	1	相机设备 ID
Width	int	384	分辨率宽度
Height	int	288	分辨率高度
VideoMode	string	"TMP"	视频模式：TMP/Y16/YUV
Fps	int	30	帧率
Exposure	int	10000	曝光时间(微秒)
AutoExposure	bool	false	自动曝光
CaptureTimeoutMs	int	100	采集超时时间

蒙版参数 (Mask)

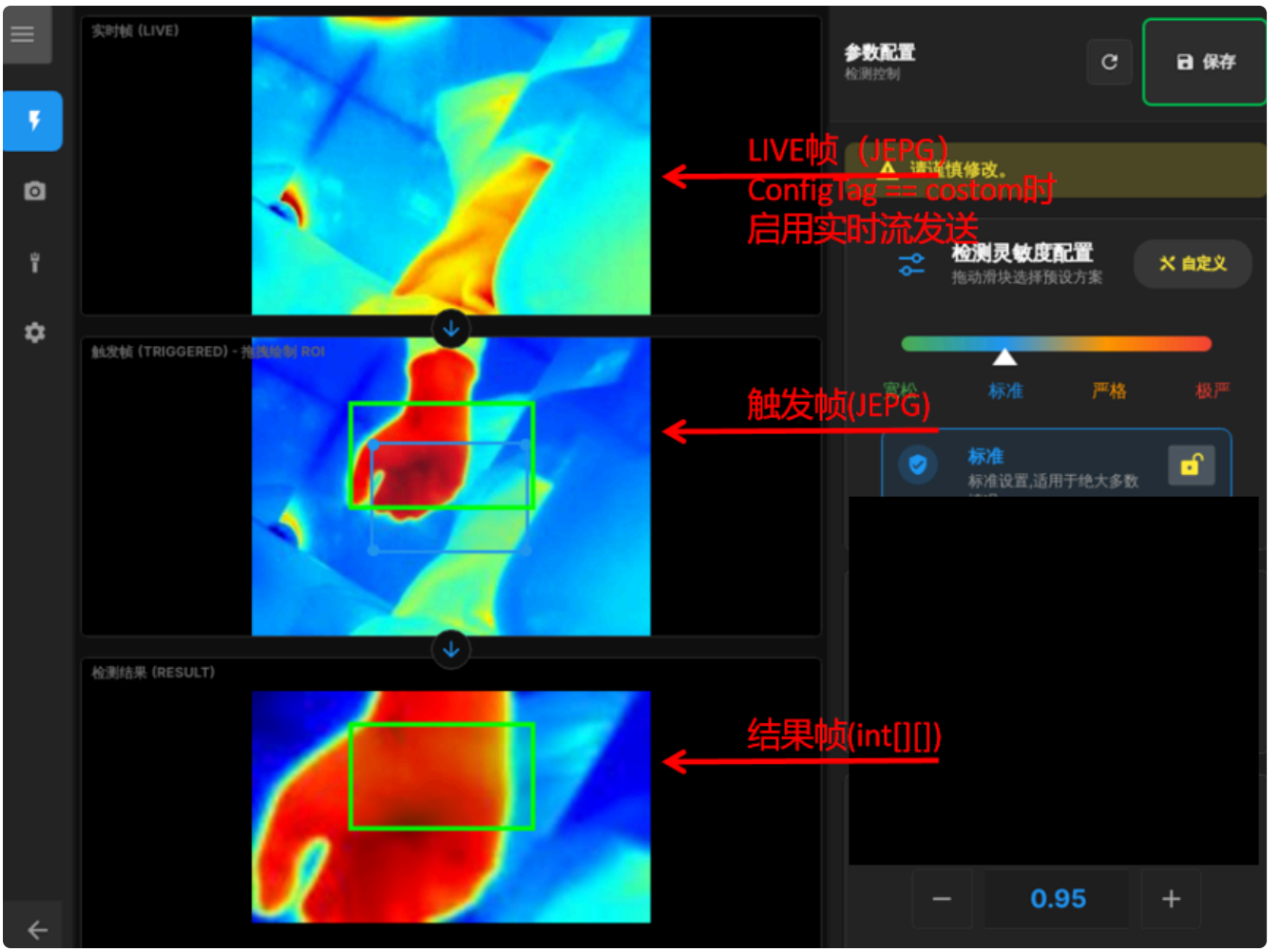
参数名	类型	默认值	说明
Enabled	bool	true	是否启用蒙版
Threshold	int	30	蒙版阈值(°C)
Width	int	185	蒙版宽度
Height	int	70	蒙版高度
Angle	int	0	旋转角度
TargetWidth	int	185	目标输出宽度
TargetHeight	int	70	目标输出高度

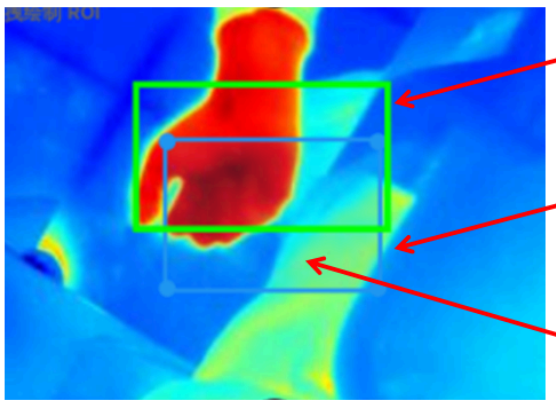
触发参数 (Trigger)

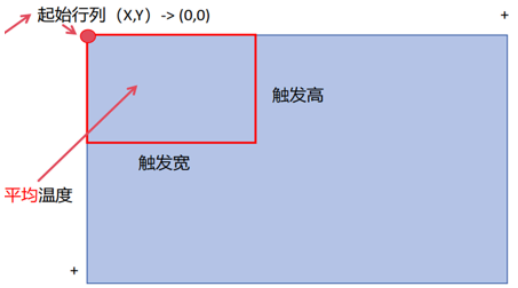
参数名	类型	默认值	说明
TriggerGpioLine	int	0	触发 GPIO 线路(0 表示禁用)
Mode	string	"External"	触发模式：Internal/External
DelayMs	int	0	触发延迟(毫秒)
BurstCount	int	1	连拍次数
InternalIntervalMs	int	100	内部触发间隔(毫秒)
TemperatureThreshold	int	45	内部触发温度阈值(°C)
DebounceIntervalMs	int	0	外部触发消抖间隔(毫秒)
TriggerRoi	Rect	0,0,0,0	触发专用 ROI 区域
TriggerCondition	string	"Average"	触发条件：Max/Average

ROI 参数

参数名	类型	默认值	说明
RegionOfInterest	Rect	0,0,160,40	算法处理边界，非必传


实时帧 (LIVE)
触发帧 (TRIGGERED) - 拖拽绘制 ROI
检测结果 (RESULT)
参数配置
检测控制
保存
请谨慎修改。
检测灵敏度配置
拖动滑块选择预设方案
自定义
宽松
标准
严格
极严
标准
标准设置,适用于绝大多数
0.95
LIVE帧（JEPG）
ConfigTag == costom时
启用实时流发送
触发帧(JEPG)
结果帧(int[][])


蒙版ROI（w,h）
X、Y计算得到
触发ROI（x,y,w,h）
RegionOfInterest
触发区域内平均温度


起始行列（X,Y）-> (0,0)
触发高
触发宽
平均温度

二维触发方式说明

外部触发

TriggerGpioLine 收到电平变化信号(低 -> 高)

触发中断 / 定时滤波 DebounceIntervalMs

DelayMs 后开始采集 , 连续采集 BurstCount 张

对截取图像进行阈值过滤 , 低于 Threshold 的值 , 被替换为 0

使用**积分图**计算 ,TargetWidth * TargetHeight 的最佳位置(X,Y)

返回被 TargetWidth * TargetHeight 裁剪的**原始图像** , 写入 int[][]中

数组打包后进入消息发送队列

内部触发

采集流程始终进行

计算图像中 TriggerRoi 中的 Max/Average, 若大于 TemperatureThreshold, 则发生内部触发

DelayMs 后开始采集 , 连续采集 BurstCount 张 , 每次采集间隔至少 InternalIntervalMs 毫秒

对截取图像进行阈值过滤 , 低于 Threshold 的值 , 被替换为 0

使用**积分图**计算 ,TargetWidth * TargetHeight 的最佳位置(X,Y)

返回被 TargetWidth * TargetHeight 裁剪的**原始图像** , 写入 int[][]中

数组打包后进入消息发送队列

---

5. 数据返回协议

5.1 数据帧结构

采集端返回数据采用 PUB-SUB 模式:

Frame 0: Topic (数据主题)

Frame 1: Data (序列化数据，JSON 格式)

5.2 数据主题定义

所有上报数据必须携带 `PipelineId` 以支持多设备区分。

主题	说明	数据格式	发送频率
`register`	设备注册信息	DeviceRegisterEvent	启动时一次
`result`	检测结果数据	AcquisitionResultEvent	每 3 秒（长帧）
`result`	检测状态	AcquisitionStatusEvent	每 3 条长帧后（短帧）
`changed`	处理完成回执	ChangedEvent	命令处理后
`status`	状态信息	StatusEvent	每 5 秒
`acquisition.frame.triggered`	触发帧数据	AcquisitionFrameEvent	触发时
`acquisition.frame.masked`	蒙版帧数据	AcquisitionFrameEvent	每帧
`acquisition.status`	采集状态	AcquisitionStatusEvent	每 1 秒

5.3 数据格式定义

5.3.1 DeviceRegisterEvent（设备注册）

设备启动时发送的注册信息。

字段定义:

JSON

```
{
  "PipelineId": "逻辑管线ID",
  "deviceId": "设备唯一标识符(UUID)",
  "sessionRuntimeHours": 0,
  "totalRuntimeHours": 0,
  "version": "1.0.0",
  "status": "Ready",
  "capabilities": ["PUB_SUB"],
  "preferredMode": "PUB_SUB"
}
```

5.3.3 ChangedEvent（处理完成回执）

命令处理完成后的回执。

字段定义：

JSON

```
{
  "PipelineId": "逻辑管线ID",
  "deviceId": "设备ID",
  "RequestId": "关联的请求ID",
  "changed": "setting/file/command",
  "status": "OK/ERROR",
  "sessionRuntimeHours": 0,
  "totalRuntimeHours": 0,
  "errorMessage": "错误信息（如有）"
}
```

5.3.4 AcquisitionFrameEvent（帧数据）

基于 PipelineFrameEvent 的简化版本，用于传输帧数据。

MessagePack(若采用简化帧)字段定义：

字段索引	字段名	类型	说明	必选
0	Version	int	协议版本(2)	✓
1	Timestamp	long	时间戳(毫秒)	✓
2	PipelineId	string	管线 ID	✓
3	FrameNumber	long	帧序号	✓
4	Width	int	图像宽度	✓
5	Height	int	图像高度	✓
6	ImageData	byte[]	图像数据(JPEG)	✗
7	MinTemperature	int	全图最低温度(°C)	✓
8	MaxTemperature	int	全图最高温度(°C)	✓
9	AvgTemperature	int	全图平均温度(°C)	✓
10	RoiTemperature	int	ROI 区域温度(°C),用于提示当前触发区域温度	✓
11	FrameType	string	帧类型	✓
12	EventId	string	事件 ID(UUID)	✓
13	Format	string	数据格式	✓
14	RoiRegion	int[]	ROI 区域[x,y,w,h]	✗
15	TemperatureMatrix	int[][]	原始温度矩阵。格式：`[ [row1], [row2], ... ]`。外层代表行，内层代表列。数据为整数（实际温度 * 10），例如 345 代表 34.5°C。	✗
16	Is2D	bool	维度标志位。`true`：二维矩阵数据；`false`：一维线性数据（此时高度为 1，格式为 `[[]]`）。	✓

数据格式说明:

- **温度数值**: 为了提高传输效率，`TemperatureMatrix` 中的数值采用整数表示。转换公式：`真实温度 = 整数值 / 10.0`。
- **矩阵结构**:
  - 二维模式 (`Is2D=true`): `[[r1c1, r1c2...], [r2c1, r2c2...]]`，外层数组的每个元素代表图像的一行。
  - 一维模式 (`Is2D=false`): `[[v1, v2, v3...]]`，外层数组仅包含一个元素（即高度为 1）。

JSON 示例 (二维数据):

JSON

```
{
  "Version": 2,
  "Timestamp": 1706941200000,
  "PipelineId": "line1",
  "FrameNumber": 100,
  "Width": 4,
  "Height": 3,
  "MinTemperature": 250,
  "MaxTemperature": 450,
  "AvgTemperature": 350,
  "RoiTemperature": 360,
  "FrameType": "Masked",
  "EventId": "uuid-123",
  "Format": "JSON",
  "Is2D": true,
  "TemperatureMatrix": [
    [250, 260, 270, 280],
    [300, 310, 320, 330],
    [400, 410, 420, 450]
  ]
}
```

JSON 示例 (一维数据):

JSON

```json
{
  "Version": 2,
  "Timestamp": 1706941200000,
  "PipelineId": "line1",
  "FrameNumber": 101,
  "Width": 4,
  "Height": 1,
  "MinTemperature": 250,
  "MaxTemperature": 300,
  "AvgTemperature": 275,
  "RoiTemperature": 275,
  "FrameType": "Masked",
  "EventId": "uuid-456",
  "Format": "JSON",
  "Is2D": false,
  "TemperatureMatrix": [
    [250, 260, 280, 300]
  ]
}
```

帧类型 (FrameType):

- `Triggered`: 触发时的原始帧
- `Masked`: 蒙版处理后的帧
- `Heatmap`: 热力图图像
- `Live`: 实时流帧

5.3.5 AcquisitionStatusEvent（采集状态）

采集端状态信息。

字段定义:

字段名	类型	说明
Version	int	协议版本
Timestamp	long	时间戳
PipelineId	string	管线 ID
Status	string	运行状态
Mode	string	当前模式
FrameRate	int	当前帧率
TemperatureRange	int[]	温度范围[min,max]
HardwareStatus	object	硬件状态
ErrorCount	int	错误计数
Uptime	long	运行时间(秒)

运行状态 (Status):

- `Idle`: 空闲
- `Running`: 运行中
- `Paused`: 暂停
- `Error`: 错误
- `Initializing`: 初始化中
- `Registering`: 注册中

---

6. 命令与响应格式

6.1 响应传输方式

由于 PUB-SUB 是单向通信，响应通过特定的响应主题发布：

- **成功响应**: `{deviceId}/response/success`
- **错误响应**: `{deviceId}/response/error`

6.2 响应格式

6.2.1 通用响应结构

响应必须包含 `RequestId` 和 `PipelineId` 以确保发送端能够正确匹配异步结果。

JSON

```
{
  "Version": 2,
  "Timestamp": 1706941200000,
  "RequestId": "对应请求ID",
  "PipelineId": "逻辑管线ID",
  "Success": true,
  "ErrorCode": 0,
  "ErrorMessage": null,
  "CommunicationMode": "PUB_SUB",
  "Data": {
    "Message": "Success"
  }
}
```

6.2.2 命令响应示例

ConfigureAcquisition 响应：

JSON

```
{
  "RequestId": "req_config_001",
  "Success": true,
  "ErrorCode": 0,
  "ErrorMessage": null,
  "CommunicationMode": "PUB_SUB",
  "Data": {
    "PipelineId": "line1",
    "AppliedParameters": ["Basic", "Camera", "Mask", "Trigger"],
    "Timestamp": 1706941200000
  }
}
```

GetStatus 响应:

JSON

```
{
  "RequestId": "req_status_001",
  "Success": true,
  "ErrorCode": 0,
  "ErrorMessage": null,
  "CommunicationMode": "PUB_SUB",
  "Data": {
    "PipelineId": "line1",
    "Status": "Running",
    "Mode": "Detect",
    "FrameRate": 29.5,
    "TemperatureRange": [20.5, 45.3],
    "Uptime": 3600,
    "ErrorCount": 0
  }
}
```

6.3 异步响应处理

由于 PUB-SUB 是异步通信，需要以下机制确保可靠性：

1. **请求 ID 匹配**: 每个命令包含唯一 RequestId，响应中携带相同 ID
2. **超时机制**: 发送命令后等待响应，超时后重试
3. **重试策略**: 失败后按指数退避重试
4. **状态同步**: 定期同步设备状态，确保一致性

---

7. 错误处理与状态码

7.1 错误代码 definition

错误代码	错误类型	说明
2001	参数验证错误	参数格式或范围错误
2002	配置应用失败	配置无法应用到硬件
2003	硬件通信错误	与相机或 GPIO 通信失败
2004	预处理错误	蒙版处理或 ROI 裁剪失败
2005	资源不足	内存或存储空间不足
2006	状态冲突	命令与当前状态冲突
2007	超时错误	操作超时
2008	数据格式错误	数据格式不支持
2101	通信模式错误	仅支持 PUB_SUB 模式
2102	主题解析错误	PUB-SUB 主题解析失败
2103	设备注册失败	设备注册过程失败
2104	订阅失败	主题订阅失败
2105	发布失败	消息发布失败

7.2 错误响应示例

JSON

```
{
  "RequestId": "req_config_001",
  "Success": false,
  "ErrorCode": 2101,
  "ErrorMessage": "仅支持PUB_SUB通信模式",
  "CommunicationMode": "PUB_SUB",
  "Data": {
    "SupportedModes": ["PUB_SUB"],
    "RequestedMode": "PUB_SUB"
  }
}
```

7.3 错误恢复策略

1. **网络错误**: 自动重连，恢复订阅
2. **数据错误**: 丢弃错误数据，记录日志
3. **配置错误**: 回退到上次有效配置
4. **硬件错误**: 进入安全模式，报告错误

---

8. 设备注册与管理

8.1 设备注册流程

Plain Text

```
采集端                          ConfigServer
   |                                |
   | 1. 连接PUB/SUB Socket          |
   |------------------------------->|
   |                                |
   | 2. 发送register消息            |
   |------------------------------->|
   | (主题: register)               |
   |                                |
   | 3. 注册设备                    |
   |<-------------------------------|
   |                                |
   | 4. 发送ready信号               |
   |<-------------------------------|
   | (主题: {deviceId}/ready)       |
   |                                |
   | 5. 设备激活，开始通信          |
   |                                |
```

8.2 设备状态管理

ConfigServer 维护设备状态表：

字段	类型	说明
DeviceId	string	设备唯一标识符
Status	string	设备状态
LastActiveTime	datetime	最后活跃时间
SessionRuntimeHours	int	本次会话运行时长
TotalRuntimeHours	int	累计运行时长
Version	string	设备版本
Capabilities	array	设备能力列表
Subscriptions	array	已订阅主题

8.3 心跳与健康检查

1. **设备心跳**：采集端每 5 秒发送 status 消息
2. **健康检查**：ConfigServer 监控设备活跃状态
3. **超时处理**：30 秒无消息视为离线
4. **自动恢复**：离线设备重新注册
