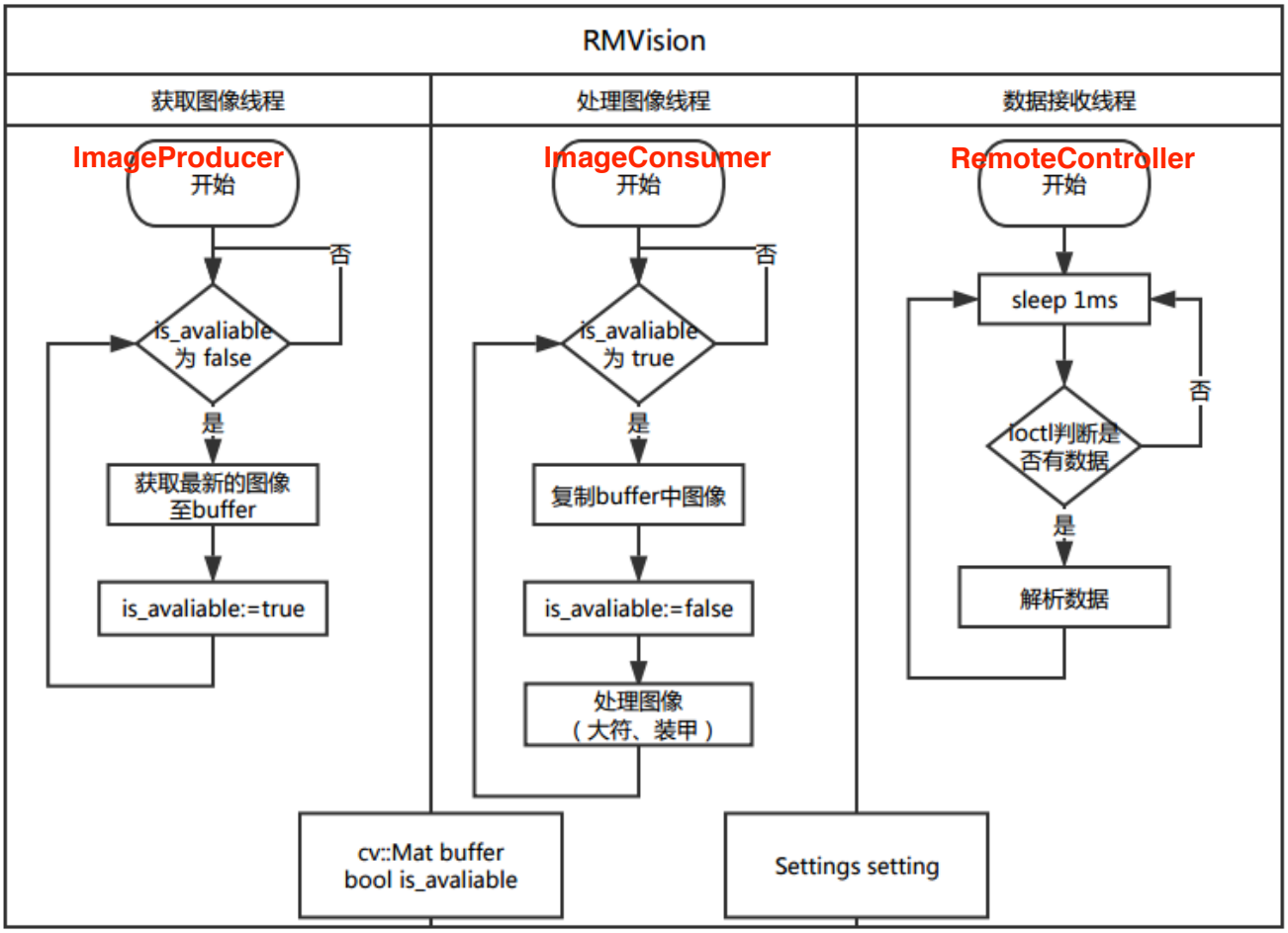
文件名	包含类	用途
AngleSolver.cpp	RectPnPSolver / AngleSolver / AngleSolverFactory	目标角度解析
ArmorDetector.cpp	ArmorParam / ArmorDetector	装甲检测
ImageConsProd.cpp	ImageConsProd	用于获取图像以及处理图像
LedController.cpp	LedController	GPIO控制LED进行状态显示
Predictor.cpp	Predictor	利用历史数据进行预测
RemoteController.cpp	RemoteController	接收其他终端数据
RMVideoCapture.cpp	RMVideoCapture	与Opencv的VideoCapture功能类似
RuneDetector.cpp	RuneDetector	神符检测
Filter.cpp	RuneResFilter / ArmorFilter / Filter1D / FilterZ	滤波器
serial.cpp		串口通信
Settings.hpp	Settings	配置文件载入器
sse_to_neon.hpp		SSE接口与NEON接口转换

系统框图



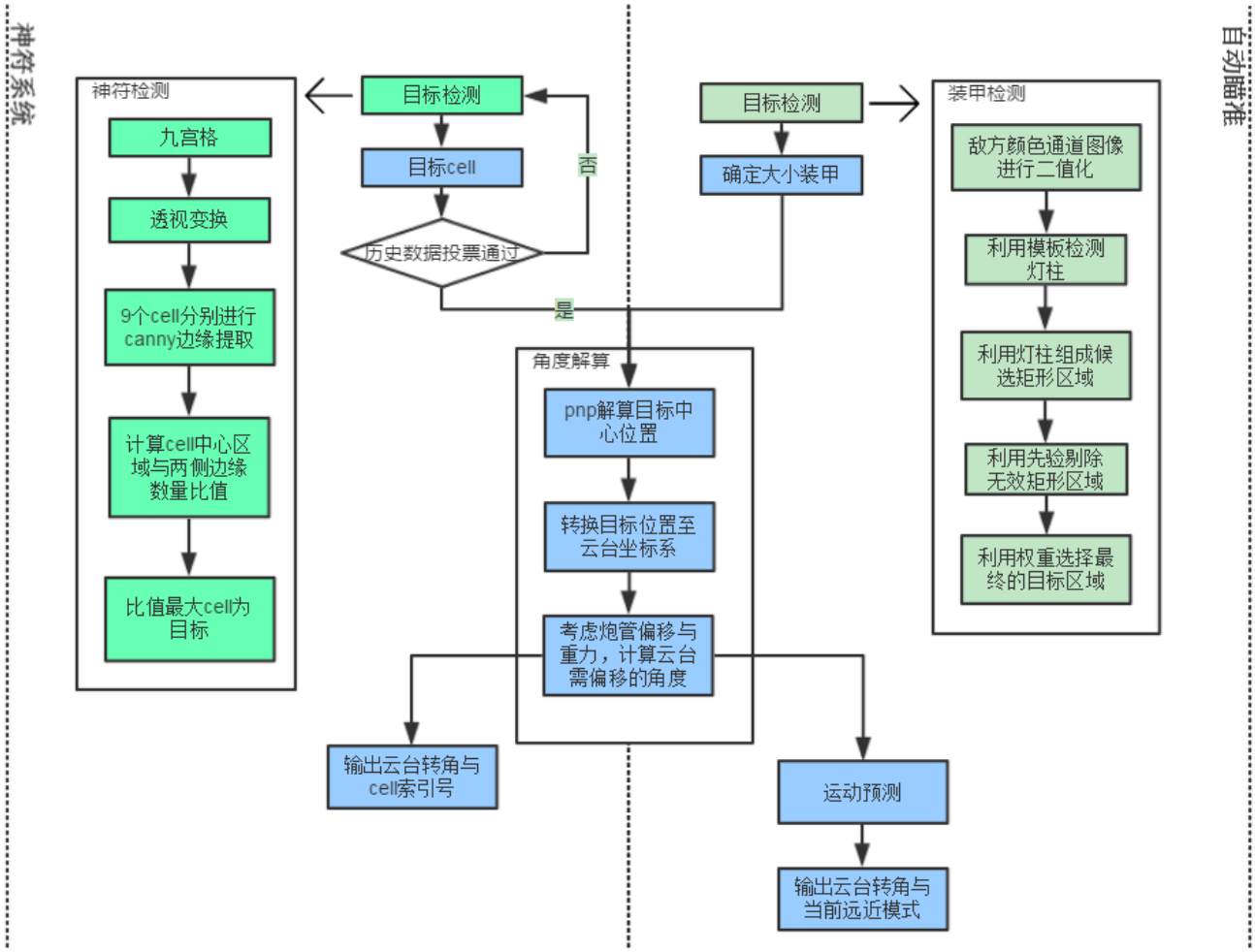
实现方案

购买的摄像头模组为KS2A17，支持

- 30fps YUV 640X480
- 120fps MJPG 640X480
- 60fps MJPG 1280X720

其中720P在垂直方向视角小于640X480，在神符模式时，对垂直方向视角要求较大，因此神符模式下选择640X480，而120fps模式下必须为低曝光，对图像的质量有一定影响，因此采用30fps YUV。而对于辅助、自动瞄准，则要求系统延时尽可能短，因此在近距离时使用120fps MJPG 640X480，而在远距离时使用60fps MJPG 1280X720，兼顾了系统的适应性与实时性。而opencv所提供的VideoCapture无法实现帧率与分辨率的选择（测试环境 ubuntu14.04 opencv3.0.0），因此采用V4l2重写了兼容opencv中Mat的RMVideoCapture类，实现了不同的分辨率与帧率的切换。

实现的总体方案如下图所示：



神符检测

基本原理

1. 二值化
2. 在二值图像中进行轮廓查找
3. 用最小的矩形包围所有轮廓，筛选出满足一定宽高比的矩形区域
4. 满足宽高比的矩形刚好等于9，则进行下一步，否则利用矩形间的距离关系选出最合适的9个作为九宫格
5. 找出九宫格的四个顶点，利用透视变换将图像变为正视图，并截取出9个cell对应的图像
6. 找出9个cell中目标的位置
    - 方案1： 利用ORB特征，进行特征匹配，匹配度最低的cell为目标
    - 方案2： 利用梯度（边缘）的数量，边缘的数量最少的为目标
    - 方案3： 利用cell中心区域的边缘数量与两侧的边缘数量比值，比值最大的为目标

投票选择

为了防止错误的检测导致大符的激活失败，我们采用了投票机制，保存最近的k次检测结果，若当前的检测结果与k次投票的检测结果一致，则输出，否者不输出并更新历史数据。

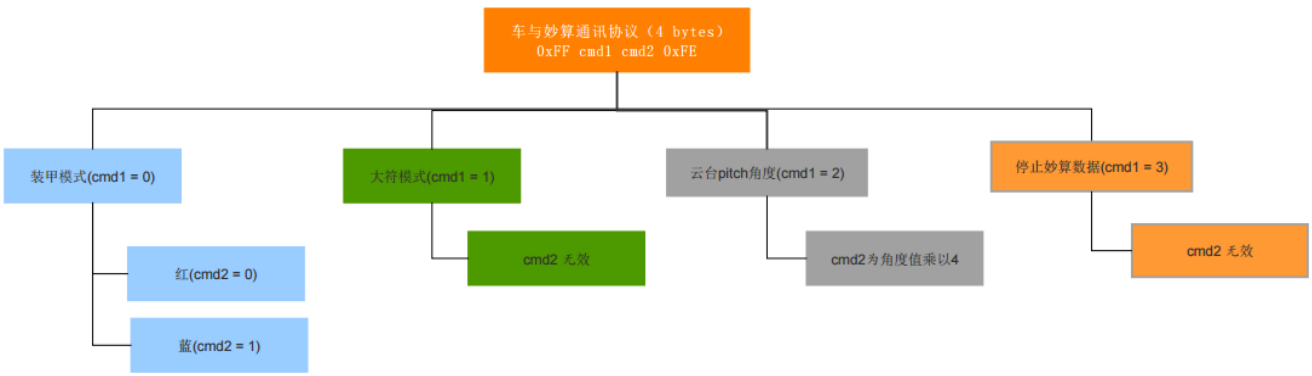
装甲检测

byte0	byte1	byte2	byte3	byte4	byte5	byte6	byte7
0xFF	data0	data1	data2	data3	data4	data5	0xFE

- 0xFF - 帧头
- [data0, data1] : 16bit int - Yaw轴角度 * 100
- [data2, data3] : 16bit int - Pitch轴角度 * 100
- [data4, data5] : 16bit int
  - 瞄准模式：
    - 步兵：（0表示数据无效，1表示当前近距离模式，2表示远距离模式）
    - 基地：目标距离（单位mm）
  - 大符模式：
    - 目标cell在九宫格中的编号（左上为1，右下为9）
- 0xFE - 帧尾

下位机至上位机协议

byte0	byte1	byte2	byte3
0xFF	cmd1	cmd2	0xFE



配置与调试

配置：

由于需要将程序部署至多台步兵上，而整个系统存在诸多参数需要调试，因此一个合理高效的调试方法显得十分重要。我们采用配置文件的方式，将整个参数放入配置文件中，配置文件如下：

```
<?xml version="1.0"?>
<opencv_storage>
<!--Parameter for Debug-->
<show_image>1</show_image>
<save_result>0</save_result>

<!--Parameter for Rune System-->
<sudoku_cell_width>66</sudoku_cell_width>
<sudoku_cell_height>37</sudoku_cell_height>
<shoot_filter_size>7</shoot_filter_size>

<!--Parameter for Armor Detection System-->
<min_light_gray>170</min_light_gray>
<min_light_height>5</min_light_height>
<avg_contrast_threshold>110</avg_contrast_threshold>
<light_slope_offset>30</light_slope_offset>
<max_light_delta_h>280</max_light_delta_h>
<min_light_delta_h>16</min_light_delta_h>
<max_light_delta_v>15</max_light_delta_v>
<max_light_delta_angle>20</max_light_delta_angle>
<avg_board_gray_threshold>60</avg_board_gray_threshold>
<avg_board_grad_threshold>25</avg_board_grad_threshold>
<grad_threshold>20</grad_threshold>
<br_threshold>0</br_threshold>

<!--Parameter for Enemy Color, 0(default) means for red, otherwise blue-->
<enemy_color>1</enemy_color>

<!--Minimum / Maximun distance (cm) of detection-->
<min_detect_distance>10.0</min_detect_distance>
<max_detect_distance>800.0</max_detect_distance>

<!--Parameter for Template-->
<template_image_file>/home/ubuntu/projects/RMVision/RMVision/template.bmp</template_image_file>
<small_template_image_file>/home/ubuntu/projects/RMVision/RMVision/small_template.bmp</small_temp
late_image_file>

<!--Parameter for Camera-->
<intrinsic_file_480>/home/ubuntu/projects/RMVision/RMVision/calibration-param/camera-RM3-04-
640.xml</intrinsic_file_480>
<intrinsic_file_720>/home/ubuntu/projects/RMVision/RMVision/calibration-param/camera-RM3-
04.xml</intrinsic_file_720>

<!--Parameter for Vision System Mode, 0(default) means for armor detection mode, 1 means for rune
system mode-->
<mode>0</mode>

<!--Bullet speed (m/s)-->
<bullet_speed>22</bullet_speed>

<!--Scale factor to adjust the distance (z-axis) obtained by PnP-->
<scale_z>1.4704</scale_z>
<scale_z_480>1.0</scale_z_480>
```

```
</opencv_storage>
```

调试：

所有的调试信息输出都将在一定程度上影响算法的实时性，因此在发布终版程序时，尽量关闭所有调试选项。调试的选项如下：

- 配置文件中`show_image`设置为1，可以看到实时的算法结果。
- 配置文件中`save_result`设置为1，将保存当前的算法结果以及原始视频流
- 将源码中的`USE_VIDEO`宏开启，可以对视频流进行逐帧调试
- 将源码中的`SHOW_DEBUG_IMG`宏开启，可以显示算法每一步的图像输出
- 将源码中的`COUT_LOG`宏开启，可以显示更多的算法细节

总结展望

总结

RMVision实现了神符系统与自动瞄准两项功能，视觉算法直接输出云台到达目标所需的转角，除了需对相机进行离线标定外，无需过多调试，有较强的硬件可移植性。

- 神符系统
  - 较强的光线适应能力
  - 较强的位置适应能力
- 自动瞄准
  - 算法简单高效，采用NEON指令集，可应对高帧率。对于640X480，可在8ms处理整图，对于1280X720可在20ms处理整图。在找到目标的情况下，由于只在小范围进行搜索，算法整体耗时非常低
  - 有效的采用若干权重来选择目标，在不减少漏检的情况下有效的减少了误检情况。
  - 有效的分辨率变化，即满足了算法的低延时，也将加大了算法的适用距离。

展望

- 由于卷帘曝光在运动过程中存在果冻效应，因此会导致存在少量的漏检，可考虑采用高帧率的全局曝光相机。
- 对于移动目标，系统的延迟主要由拨弹、出弹、子弹飞行等导致，而非检测算法。因此一个有效的预测算法对于移动目标显得更为重要。目前仅简单的采用了二次函数拟合最近的历史数据来进行目标预测，对于云台和目标都在运动的情况，效果并不是十分理想，能否结合云台Pitch、Yaw运动数据，更真实的还原目标的运动轨迹来预测目标位置进行打击，可以作为自动瞄准的一个重点研究方向。