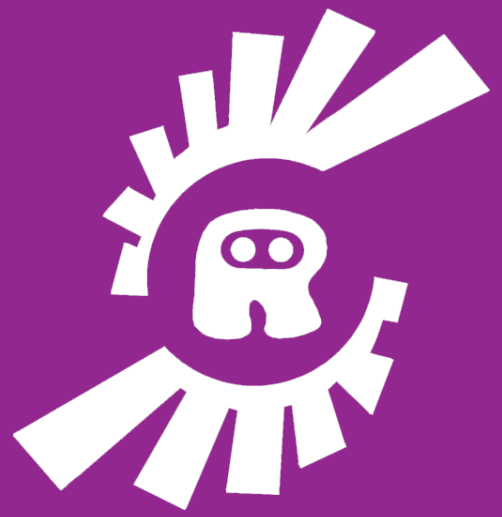
南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

视觉技术分享

汇报人：施磊

團结 敢身 求是 創新



目录

CONTENTS

南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

第一部分

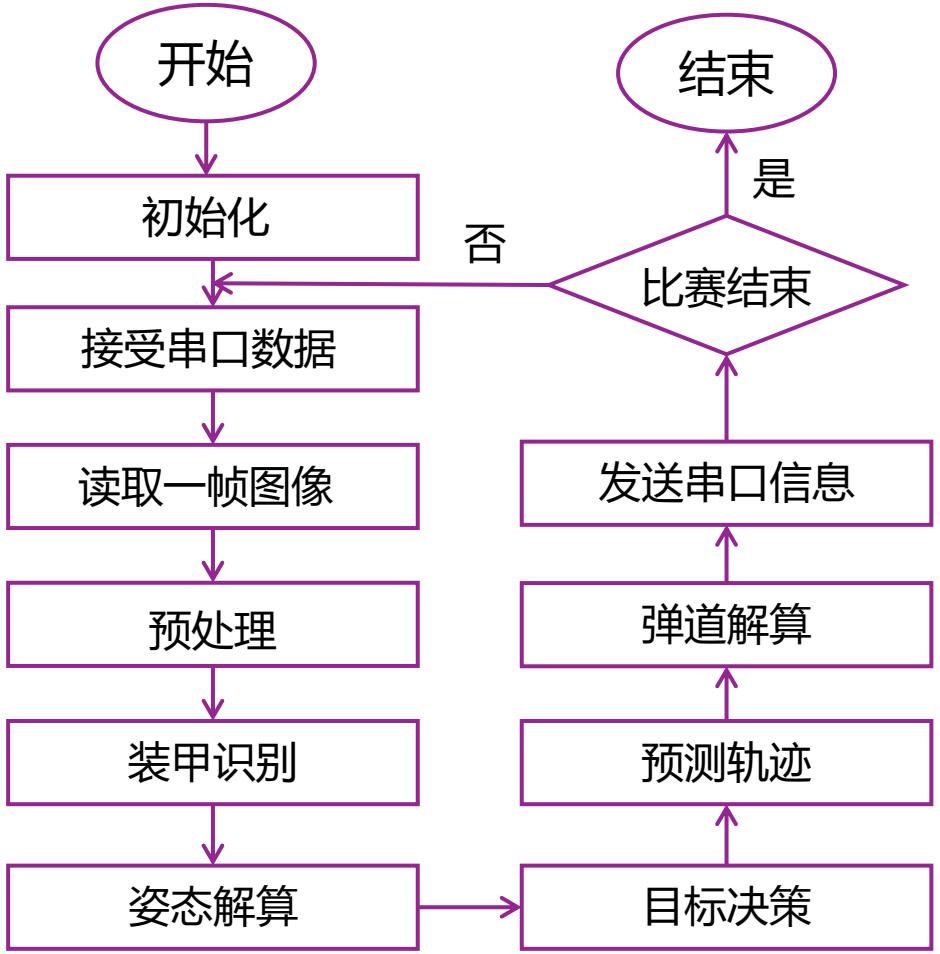
视觉算法/PART 1

装甲板识别

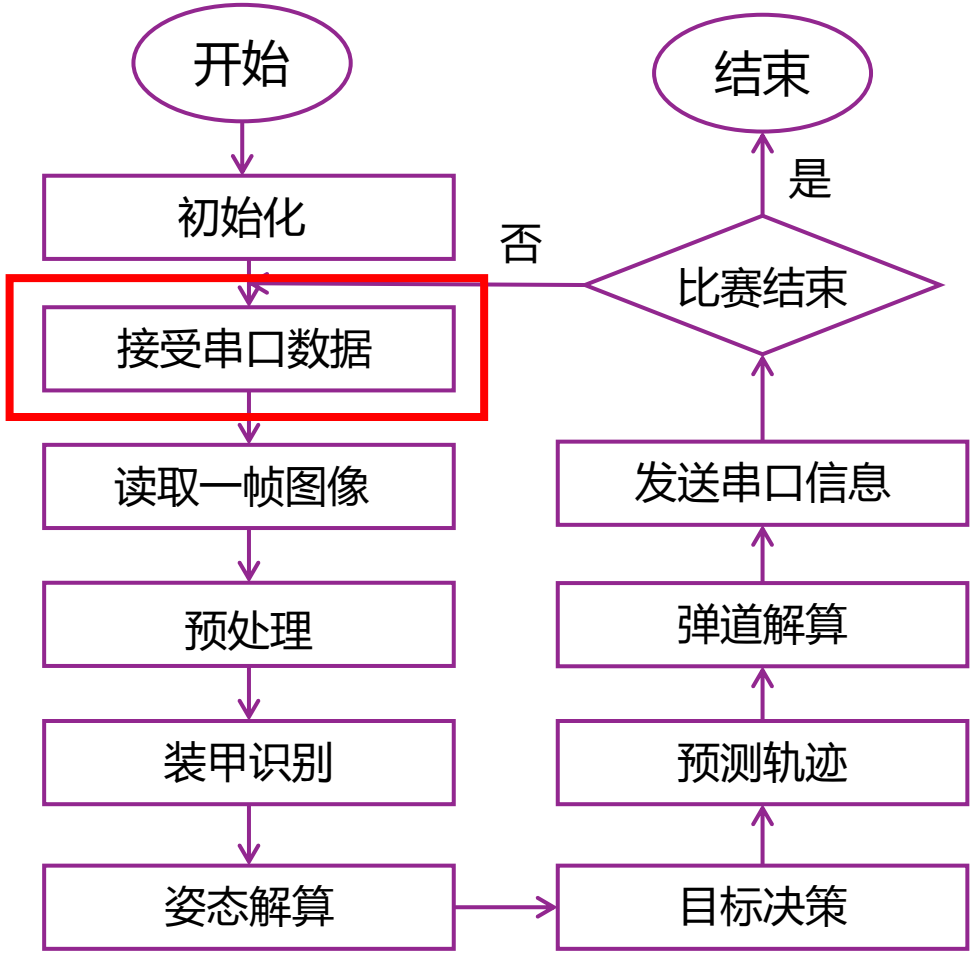
ALLIANCE

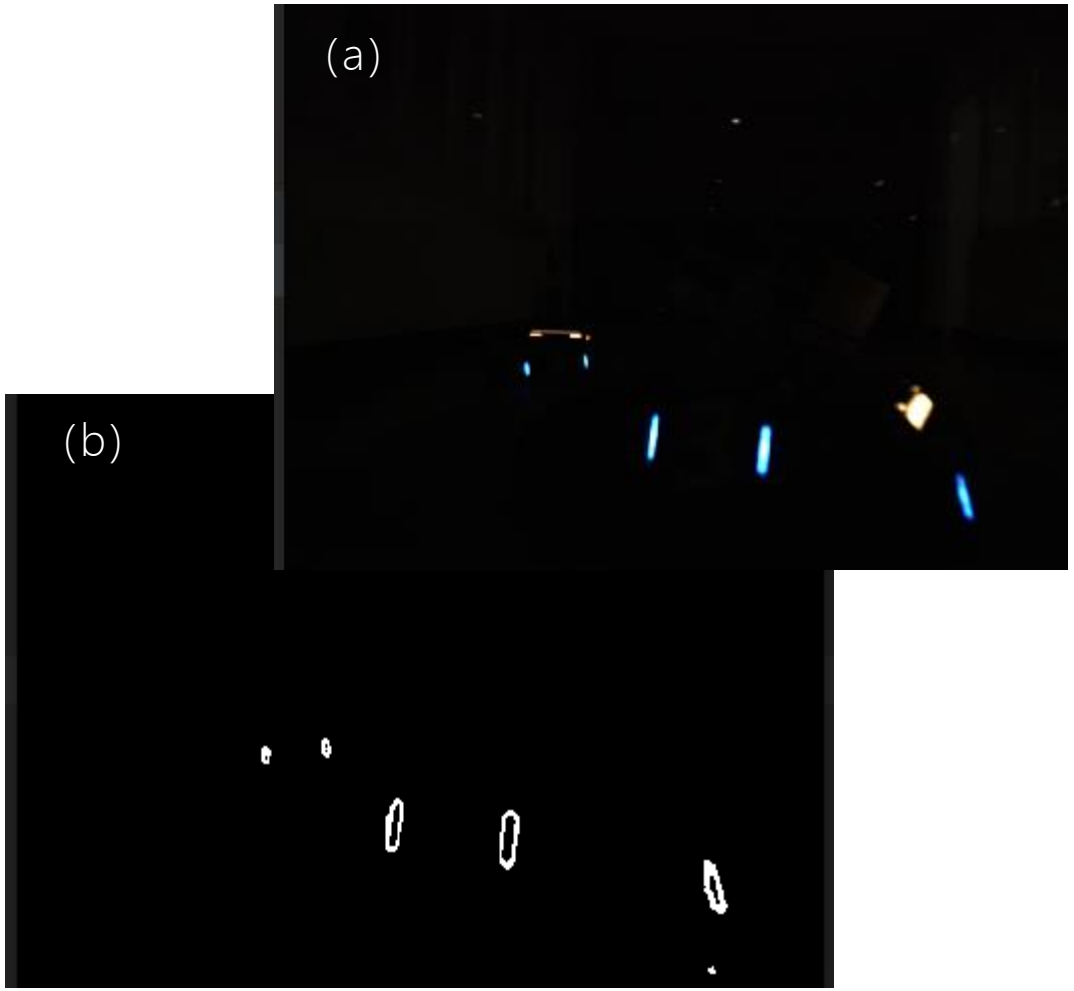
需求分析

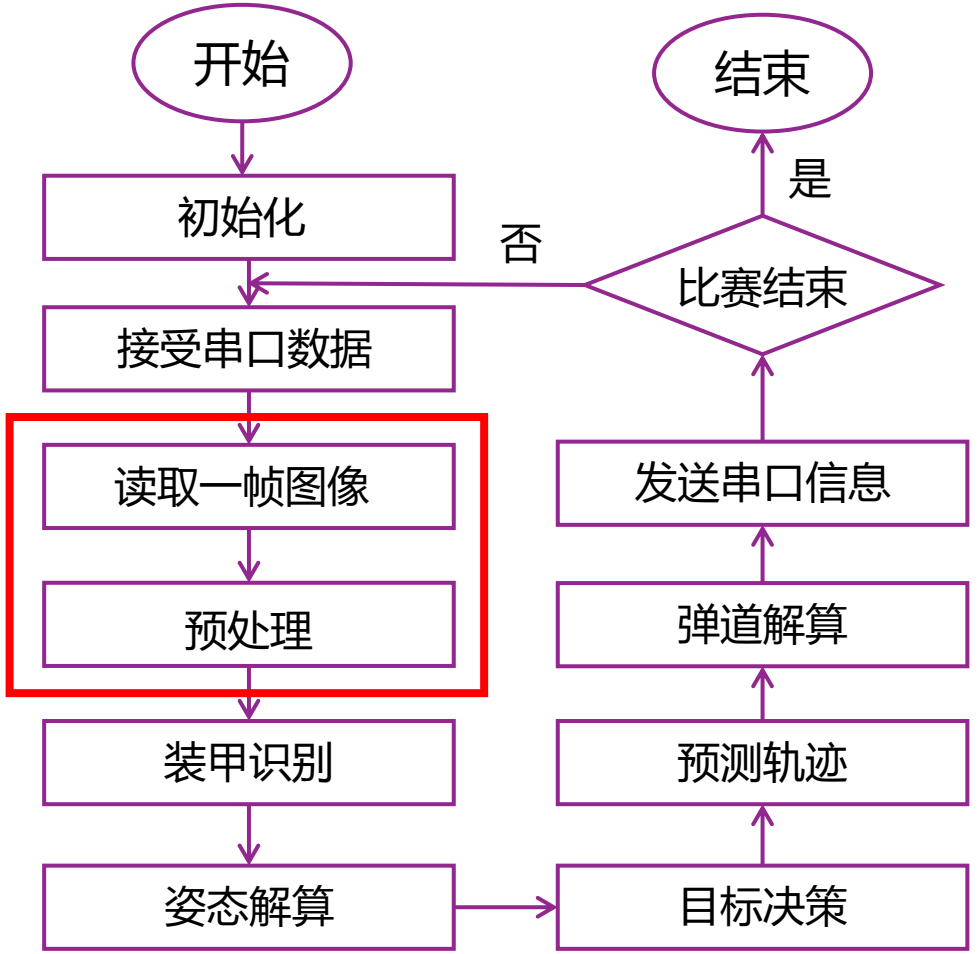
- 识别出图像中的目标装甲板
- 预测目标运动
- 计算弹道

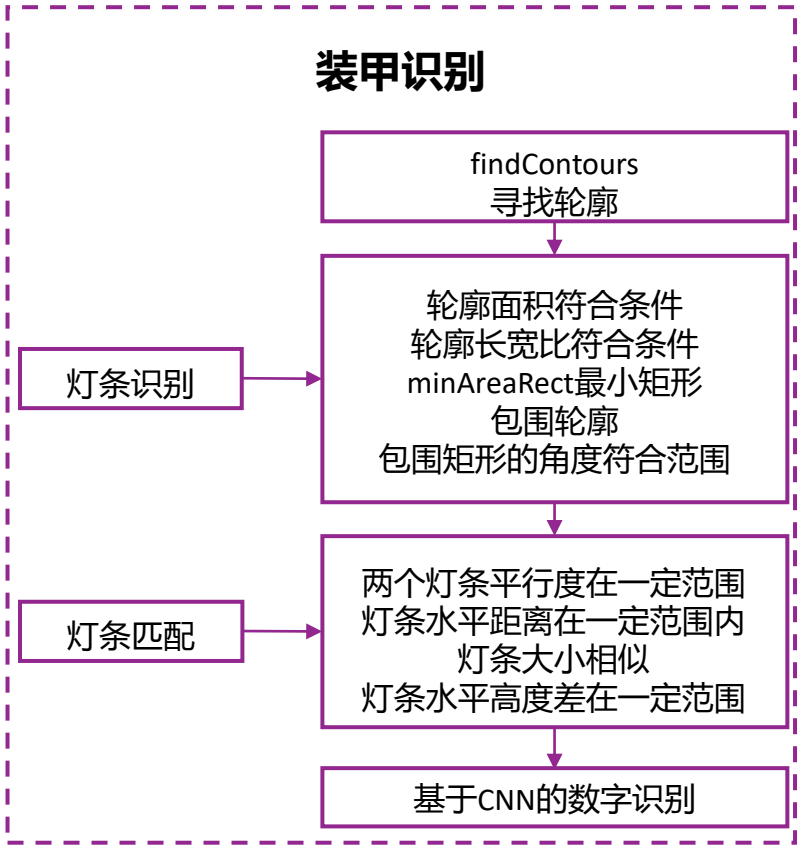


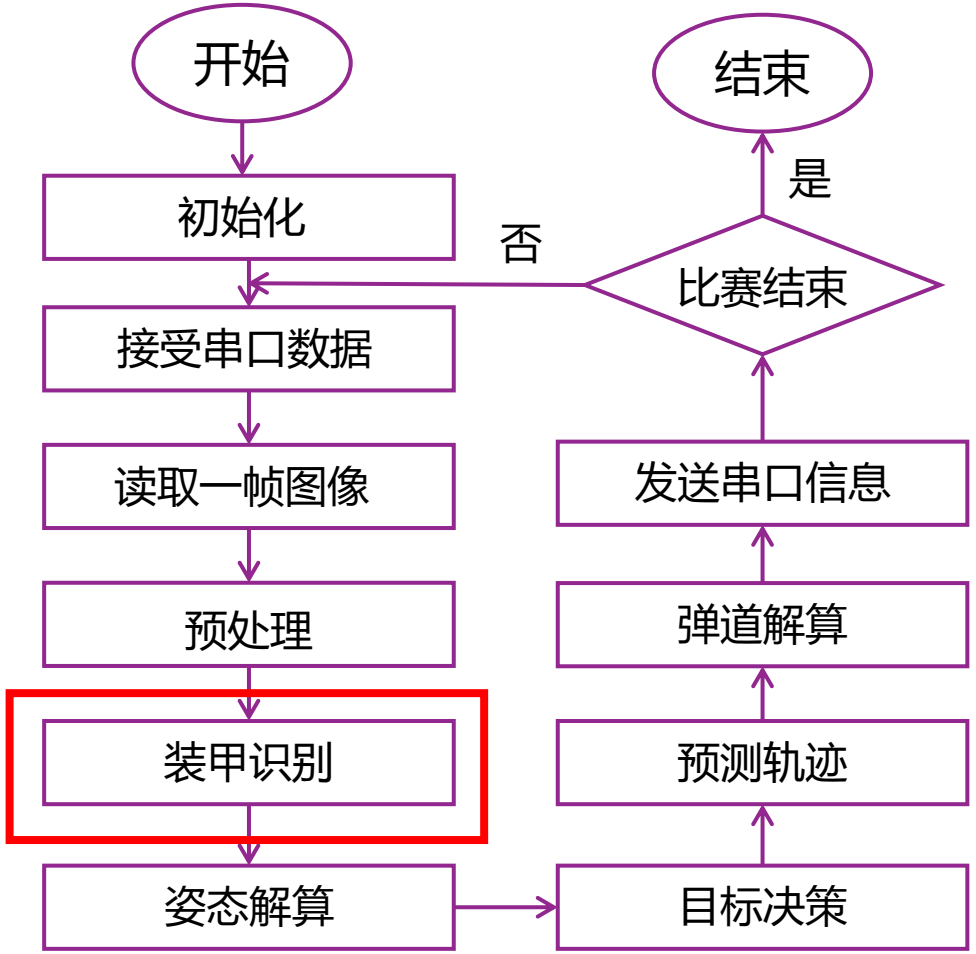
```
struct Rxpackage
{
    char head = 0xff;
    float pitch;
    float yaw;
    float roll;
    float spd;
    sint color;
    sint gryo;
    sint mode;
    float yaw_spd;
    float pitch_spd;
    char tail = 0xfe;
};
```

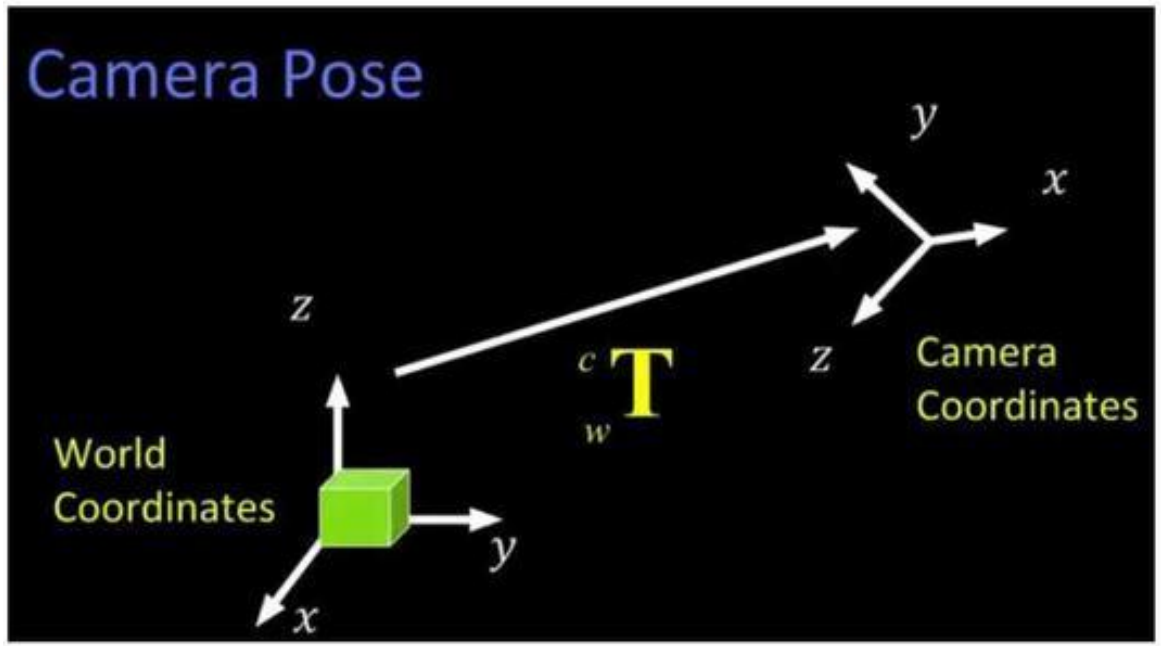



(a)
(b)

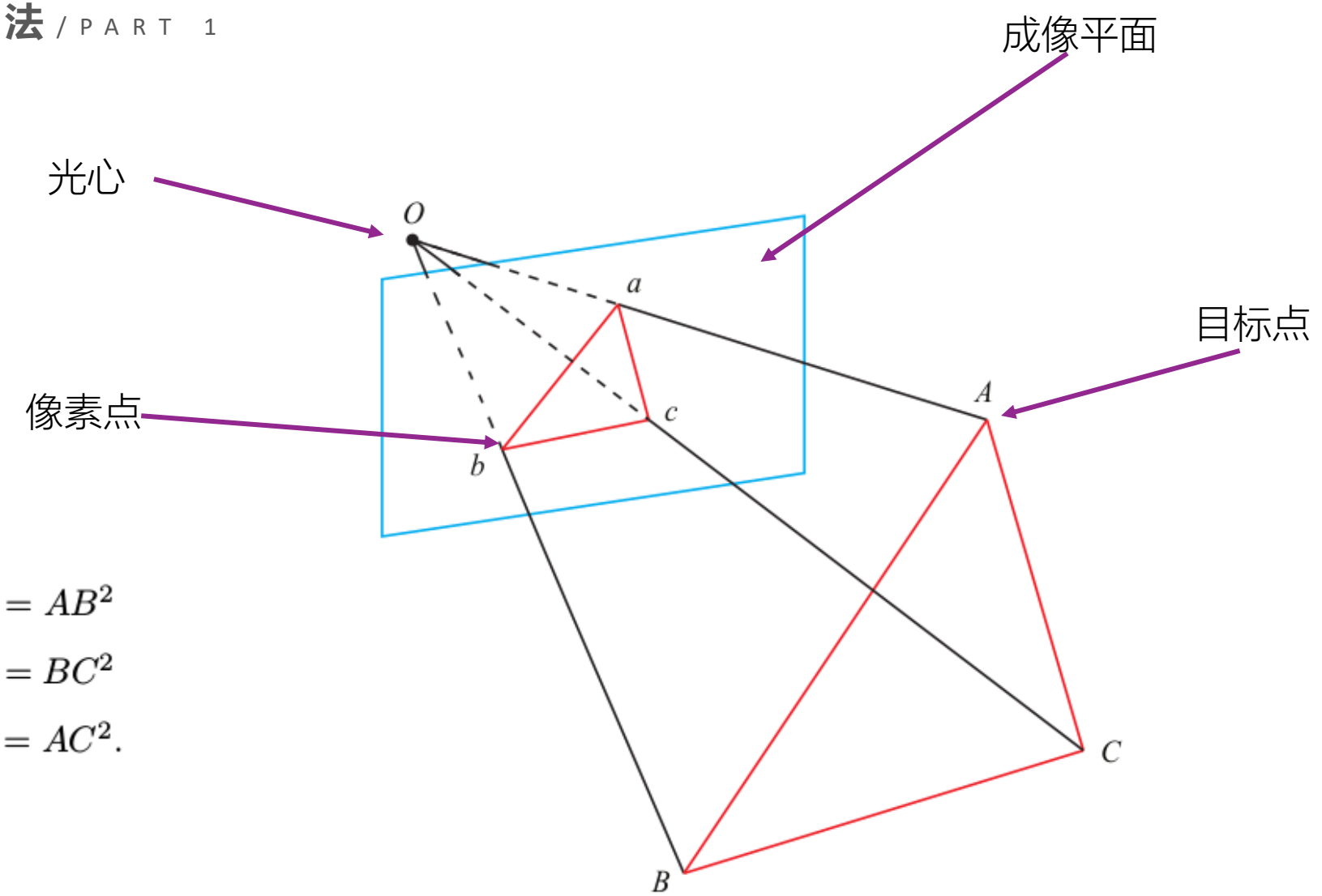

开始
初始化
接受串口数据
读取一帧图像
预处理
装甲识别
姿态解算
目标决策
预测轨迹
弹道解算
发送串口信息
比赛结束
否
是
结束


装甲识别
findContours
寻找轮廓
灯条识别
轮廓面积符合条件
轮廓长宽比符合条件
minAreaRect最小矩形
包围轮廓
包围矩形的角度符合范围
灯条匹配
两个灯条平行度在一定范围
灯条水平距离在一定范围内
灯条大小相似
灯条水平高度差在一定范围
基于CNN的数字识别


开始
初始化
接受串口数据
读取一帧图像
预处理
装甲识别
姿态解算
目标决策
预测轨迹
弹道解算
发送串口信息
比赛结束
否
是
结束


Camera Pose
y
x
z
c
T
w
z
Camera
Coordinates
World
Coordinates
y
x


开始
初始化
接受串口数据
读取一帧图像
预处理
装甲识别
姿态解算
目标决策
预测轨迹
弹道解算
发送串口信息
比赛结束
否
是
结束



$$OA^2 + OB^2 - 2OA \cdot OB \cdot \cos\langle a, b\rangle = AB^2$$
$$OB^2 + OC^2 - 2OB \cdot OC \cdot \cos\langle b, c\rangle = BC^2$$
$$OA^2 + OC^2 - 2OA \cdot OC \cdot \cos\langle a, c\rangle = AC^2.$$

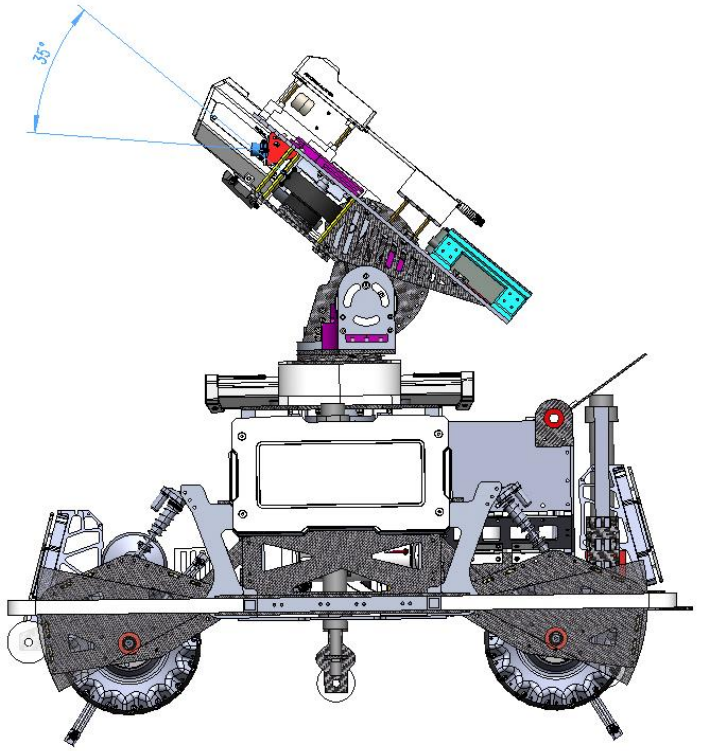
像素点　相机内参　旋转矩阵　平移矩阵　空间点

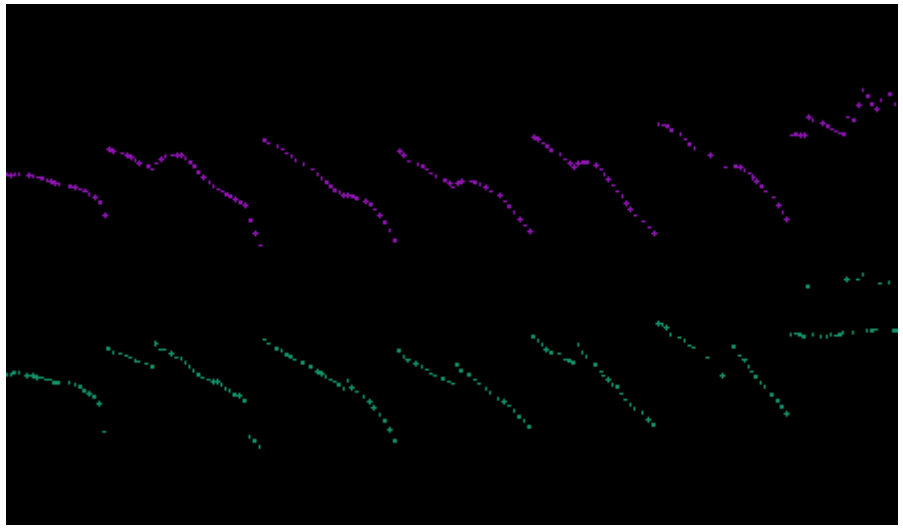
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

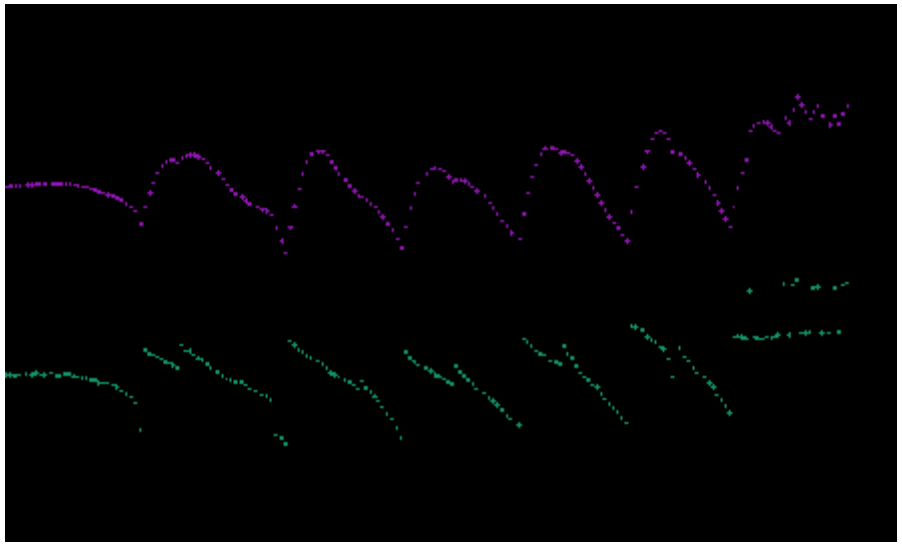
$$Rx=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(roll) & -\sin(roll)\\ 0 & \sin(roll) & \cos(roll)\end{bmatrix}$$

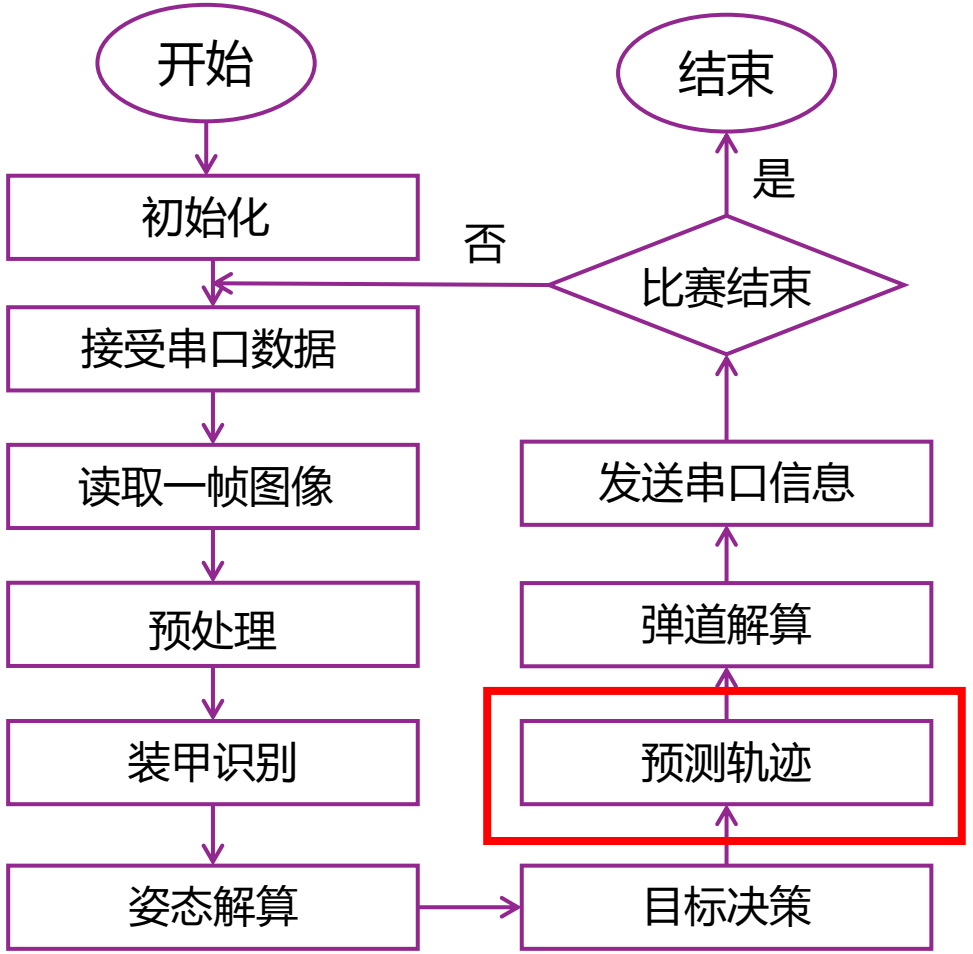
$$Ry=\begin{bmatrix}\cos(pitch) & 0 & \sin(pitch)\\ 0 & 1 & 0\\ -\sin(pitch) & 0 & \cos(pitch)\end{bmatrix}$$

$$Rz=\begin{bmatrix}\cos(yaw) & -\sin(yaw) & 0\\ \sin(yaw) & \cos(yaw) & 0\\ 0 & 0 & 1\end{bmatrix}$$

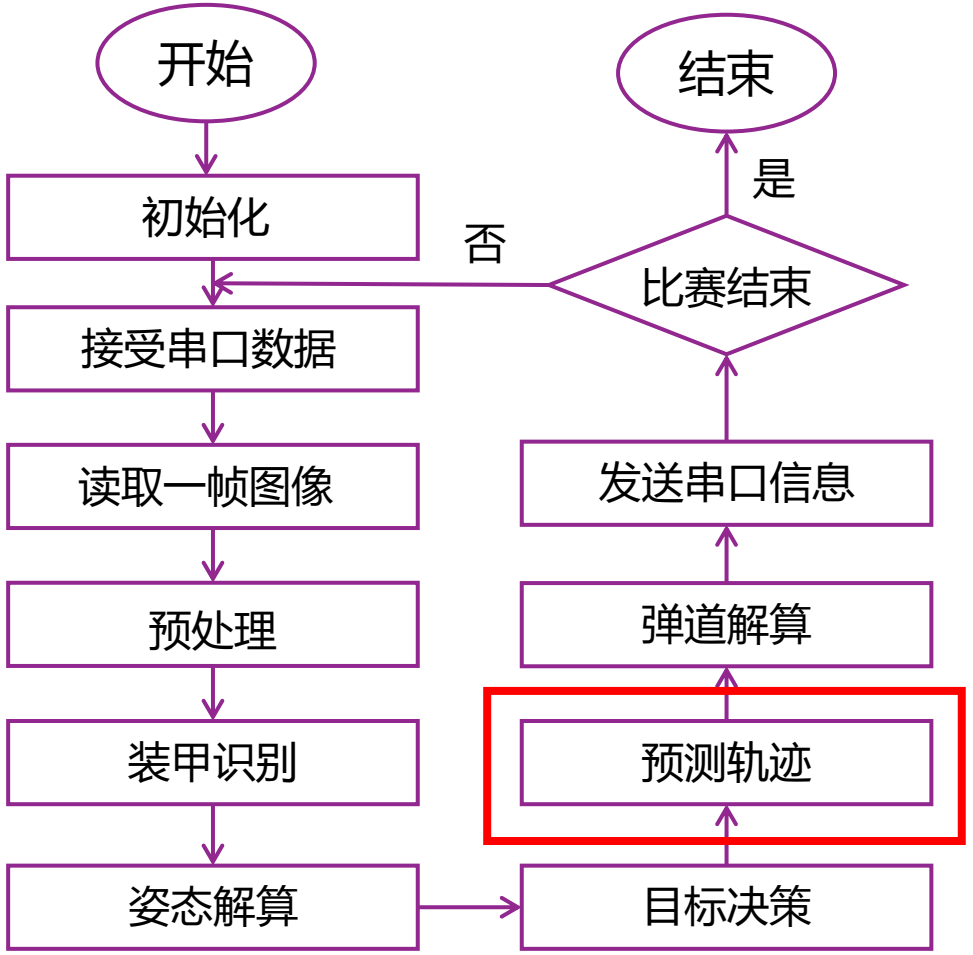


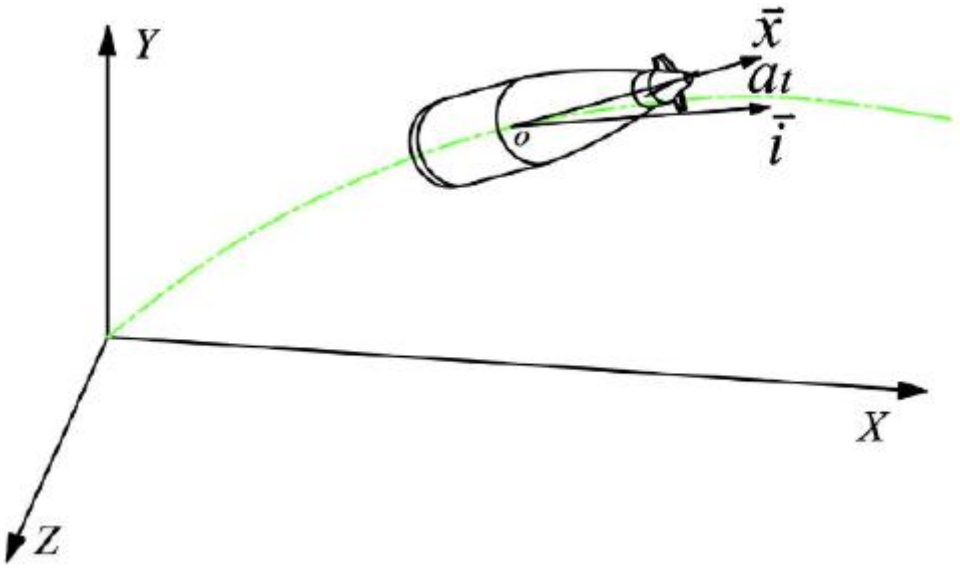


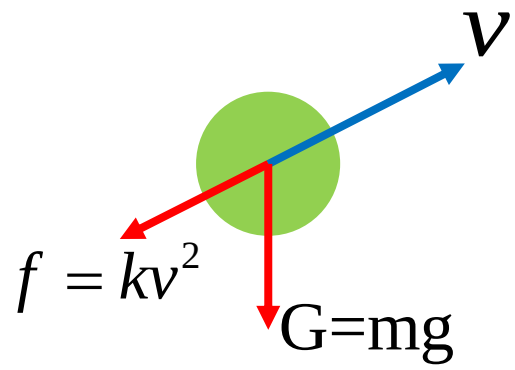


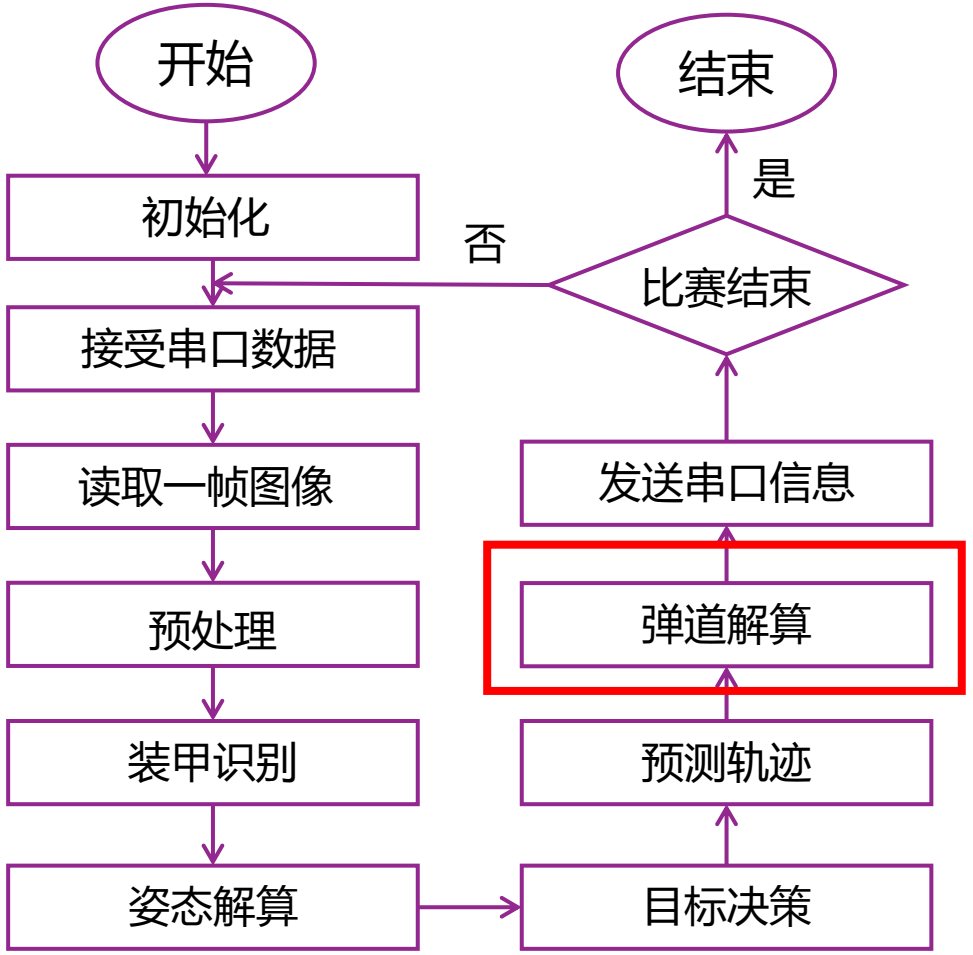

开始
初始化
接受串口数据
读取一帧图像
预处理
装甲识别
姿态解算
否
比赛结束
是
结束
发送串口信息
弹道解算
预测轨迹
目标决策

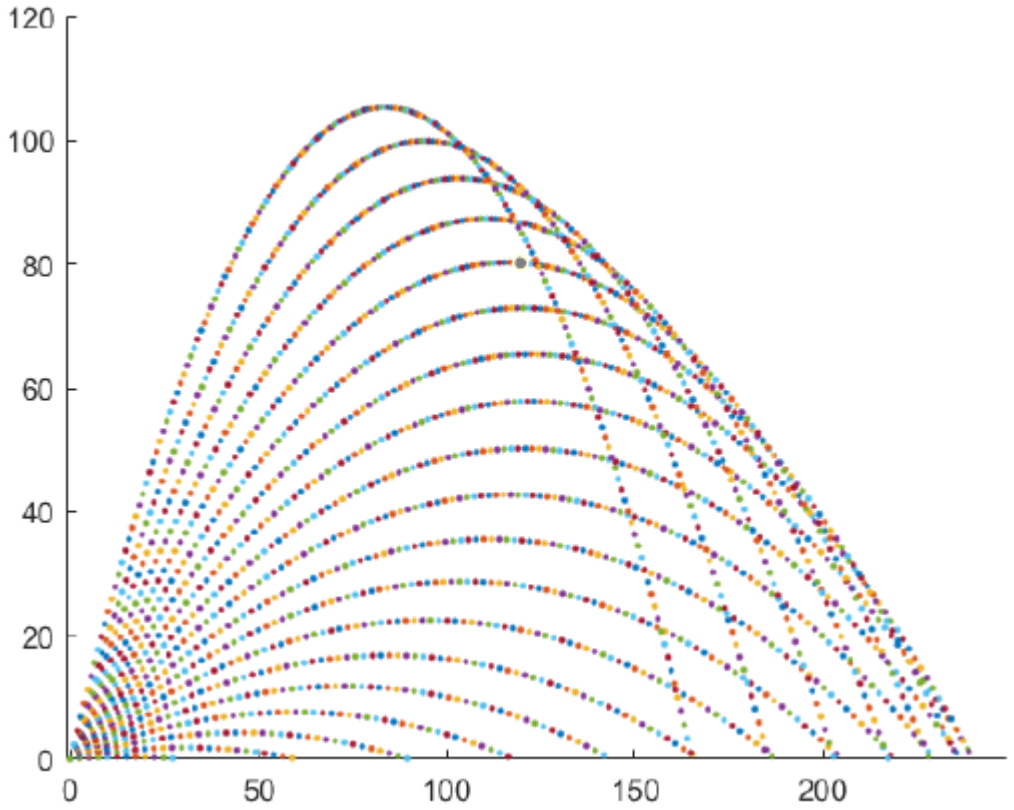
```
cv::Point3f Prophet4Armor::predict(float t, cv::Point3f &v) {
    v = cv::Point3f(kf.statePost.at<float>(3),
                    kf.statePost.at<float>(4),
                    kf.statePost.at<float>(5));
    state = cv::Point3f(kf.statePost.at<float>(0),
                                kf.statePost.at<float>(1),
                                kf.statePost.at<float>(2));

    return (kf.predict()-state)*t+state;
}
```




Y
X
Z
o
$\vec{x}$
a_t
$\vec{i}$


v
$f = kv^2$
G=mg


开始
初始化
接受串口数据
读取一帧图像
预处理
装甲识别
姿态解算
目标决策
预测轨迹
弹道解算
发送串口信息
比赛结束
否
是
结束



$$t = t_i$$

$$v_i \quad x_i \quad f_i = kv_i$$

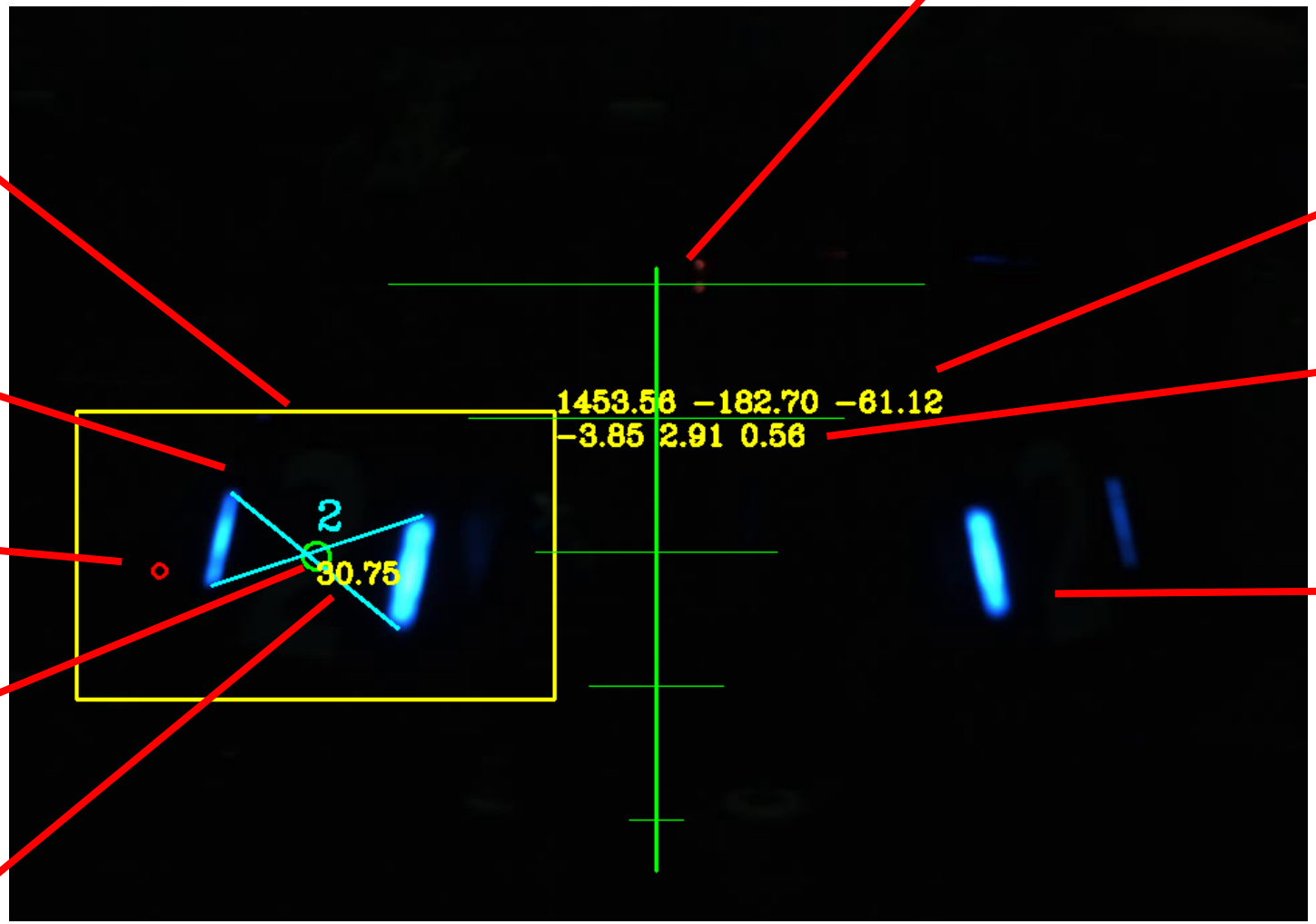
$$t = t_{i+1}$$

$$F_i = f_i + mg \qquad a = \frac{F_i}{m}$$

$$v_{i+1} = v_i + a \Delta t$$

$$x_{i+1} = x_i + v_i \Delta t$$

$$f_{i+1} = kv_{i+1}$$


辅助线，方便弹道解算参数调节
选择该装甲板作为目标
目标坐标
灯条识别结果
目标速度
1453.56 -182.70 -61.12
-3.85 2.91 0.56
预测结果
2
30.75
非目标装甲板
灯条匹配及数字识别
目标打分

南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

第二部分

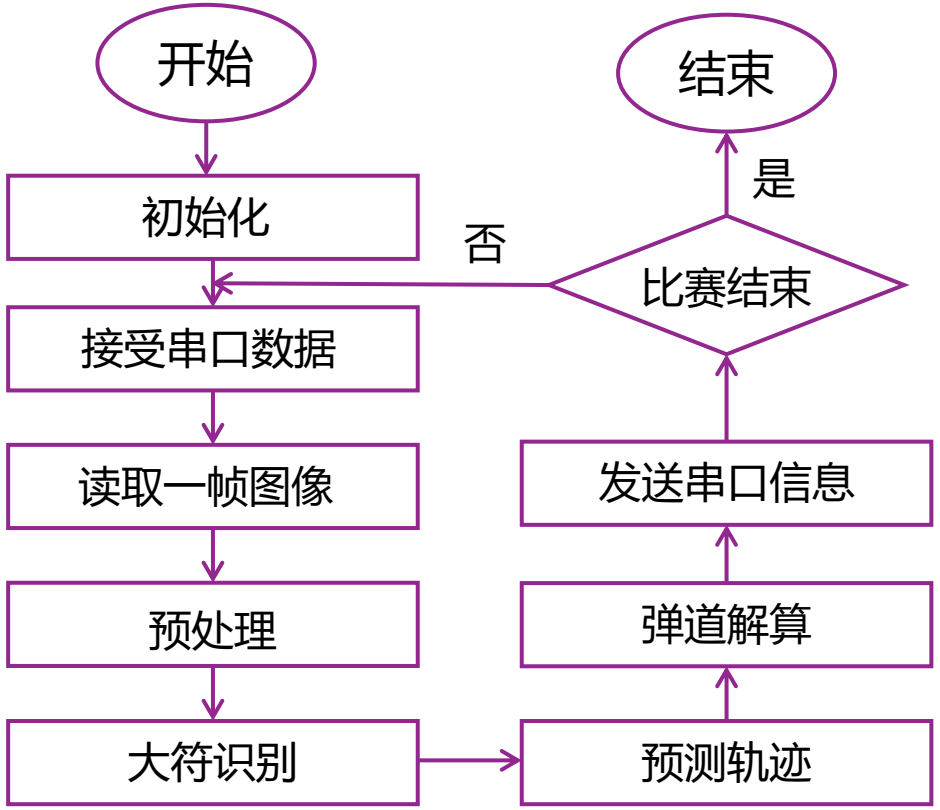
视觉算法/PART 2

大符识别

ALLIANCE

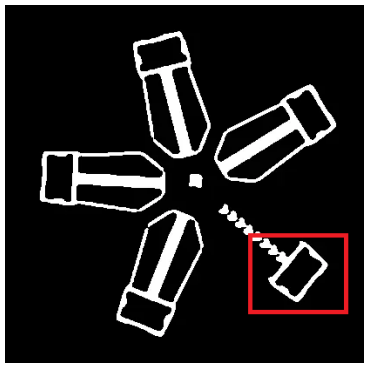
需求分析

- 识别出图像中未被打击的大符
- 预测大符的
- 计算弹道

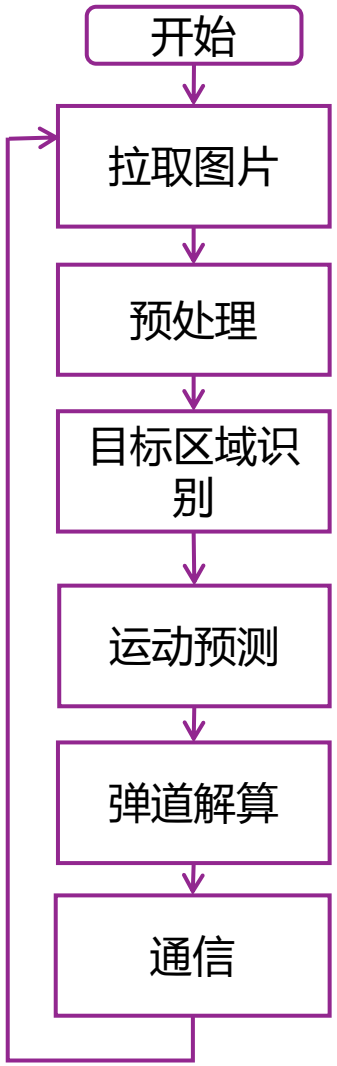


能量机关

能量机关原图



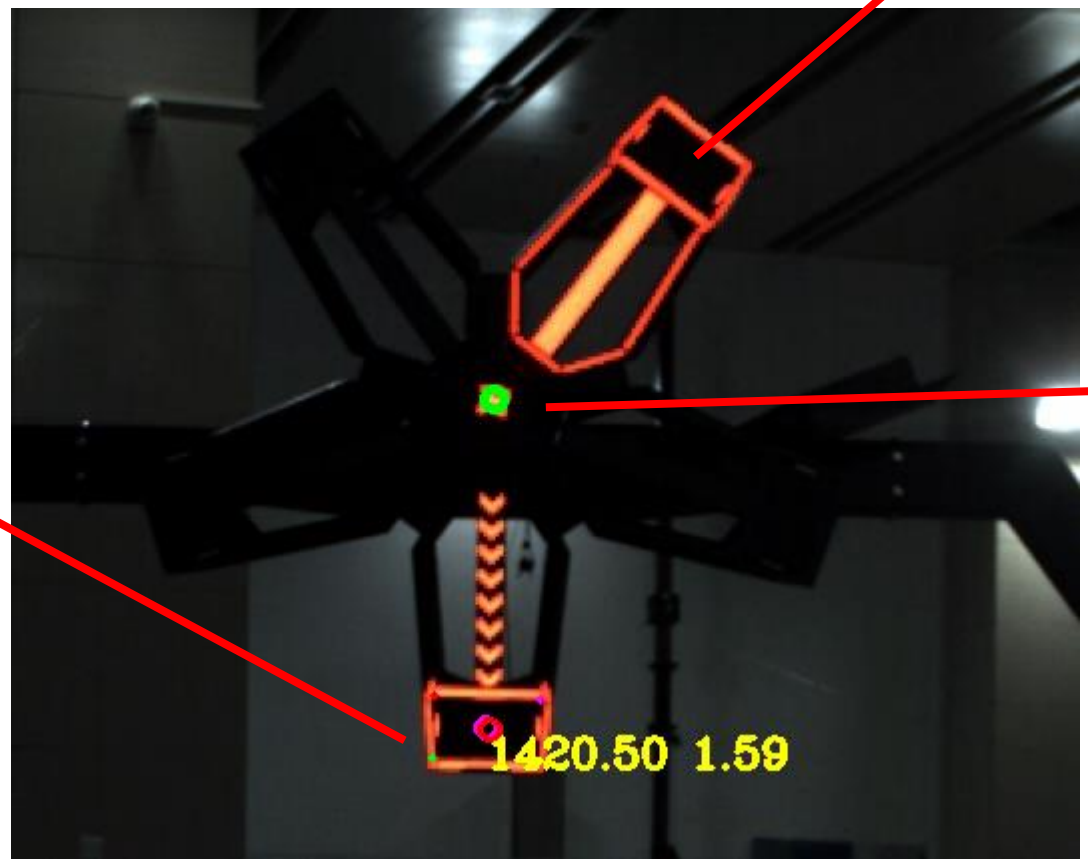
能量机关处理图



首先用findContours函数找出轮廓，并且找到二值化图像中的连通域，**根据是否为外轮廓和面积大小初步识别出扇叶和其是否被击打。**至此已经可以完成识别任务，但为保证识别效果，后续根据扇叶中是否有流动的箭头，外轮廓里**内轮廓的hu距**大小进一步做判断。

基于PnP解算可以得到世界坐标相对与相机坐标系的**旋转向量R**和平移向量T。解算云台转动角度及射击时机得到打击点目标位置后，忽略垂直的空气阻力，考虑水平空气阻力的弹道模型，解算云台转动角度。当云台转动角度小于一定值时，向下位机发击命令。

已击打，不识别



大符旋转中心

目标
区域面积：1420.50
角度：1.59（91.15°）

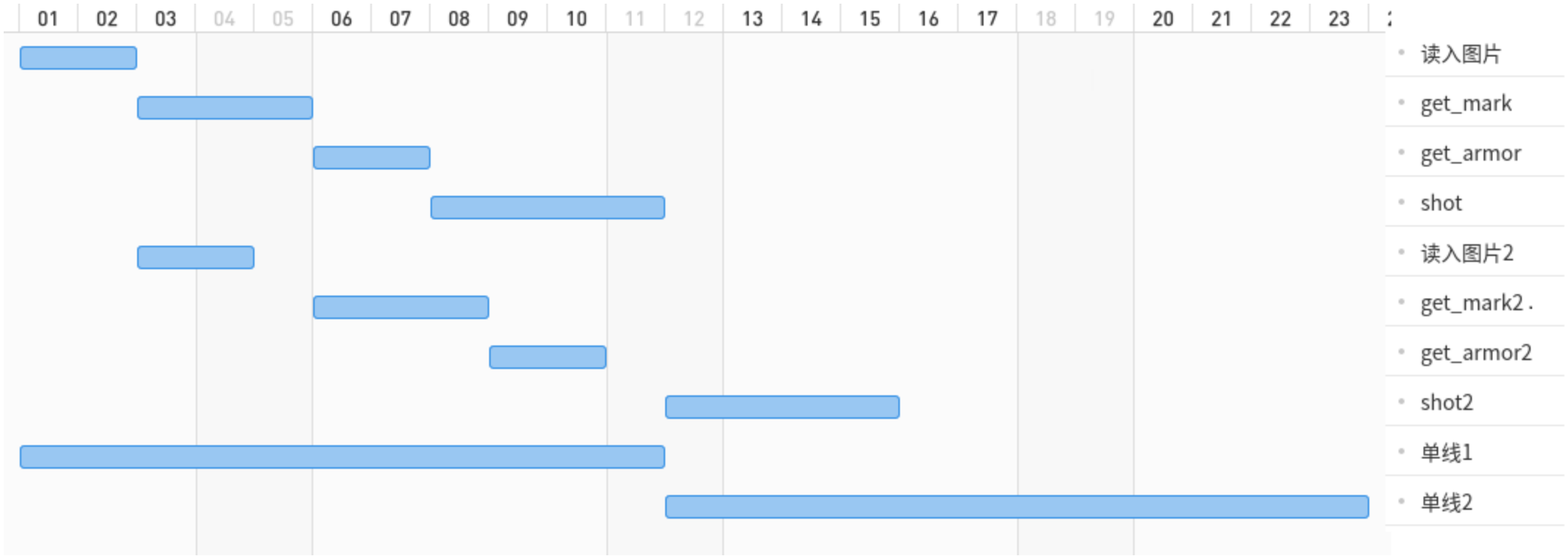
南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

第三部分

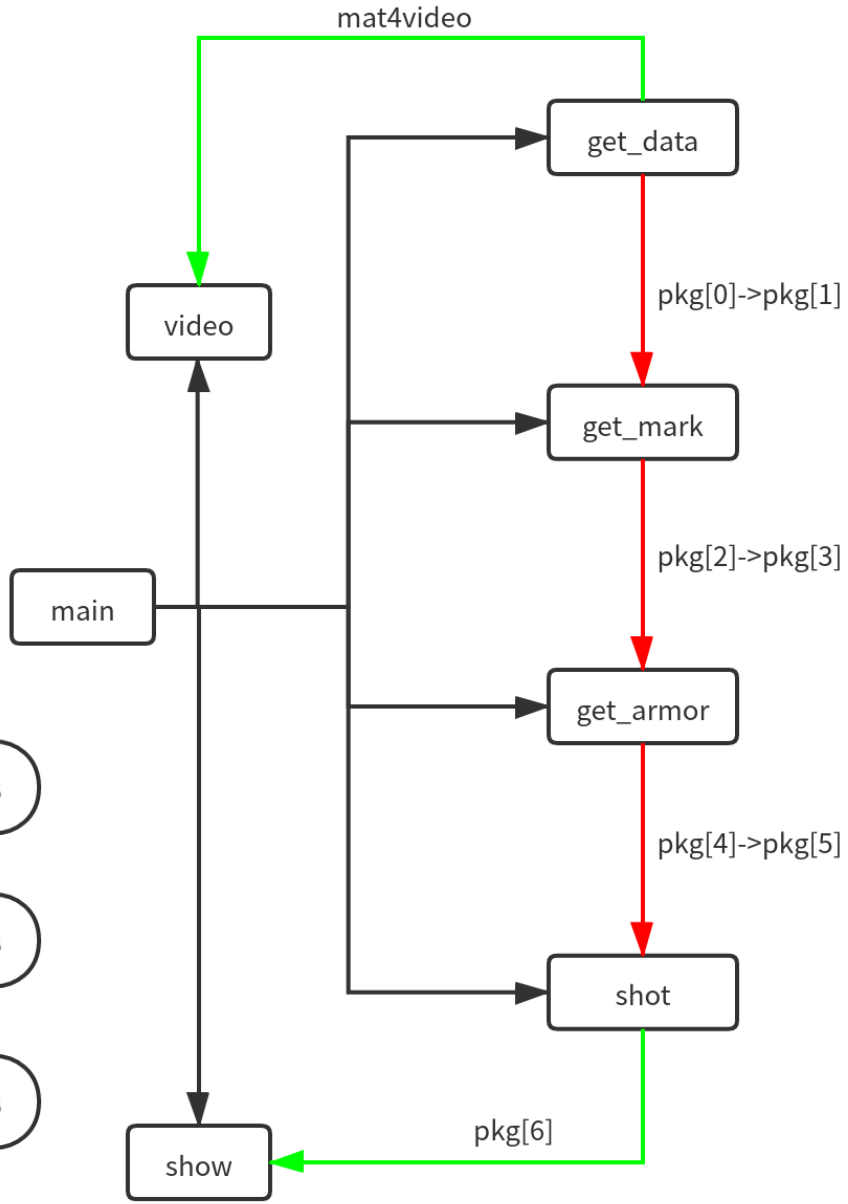
视觉算法/PART 3

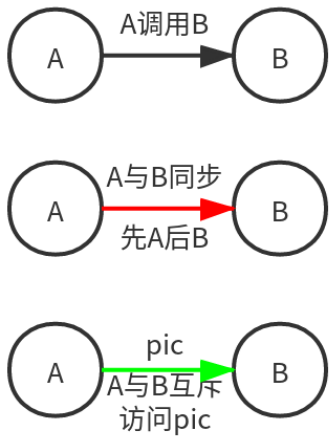
多 线 程

ALLIANCE


01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
读入图片
get_mark
get_armor
shot
读入图片2
get_mark2 .
get_armor2
shot2
单线1
单线2

```
std::thread data(get_data, std::ref(camera), std::ref(alliance));
std::thread mark(get_mark, std::ref(alliance));
std::thread armor(get_armor, std::ref(alliance));
std::thread shoth(shot, std::ref(alliance));

if(tmpset.video){
    std::thread Video(video, std::ref(alliance));
    Video.detach();
}
if(tmpset.showimg){
    std::thread showth(show, std::ref(alliance));
    showth.join();
}

shoth.join();
armor.join();
mark.join();
data.join();
```





南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

第四部分

视觉算法/PART 4

Kalman滤波

ALLIANCE

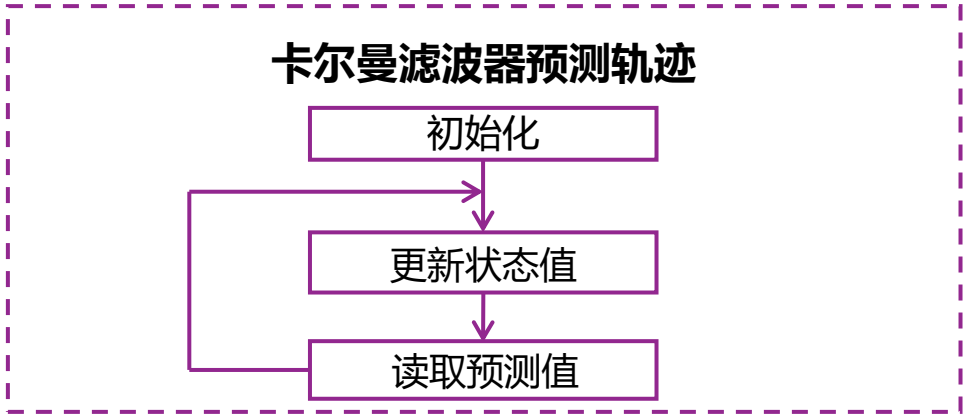
$$\begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} 1 & & & 1 & & \\ & 1 & & & 1 & \\ & & 1 & & & 1 \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix}$$

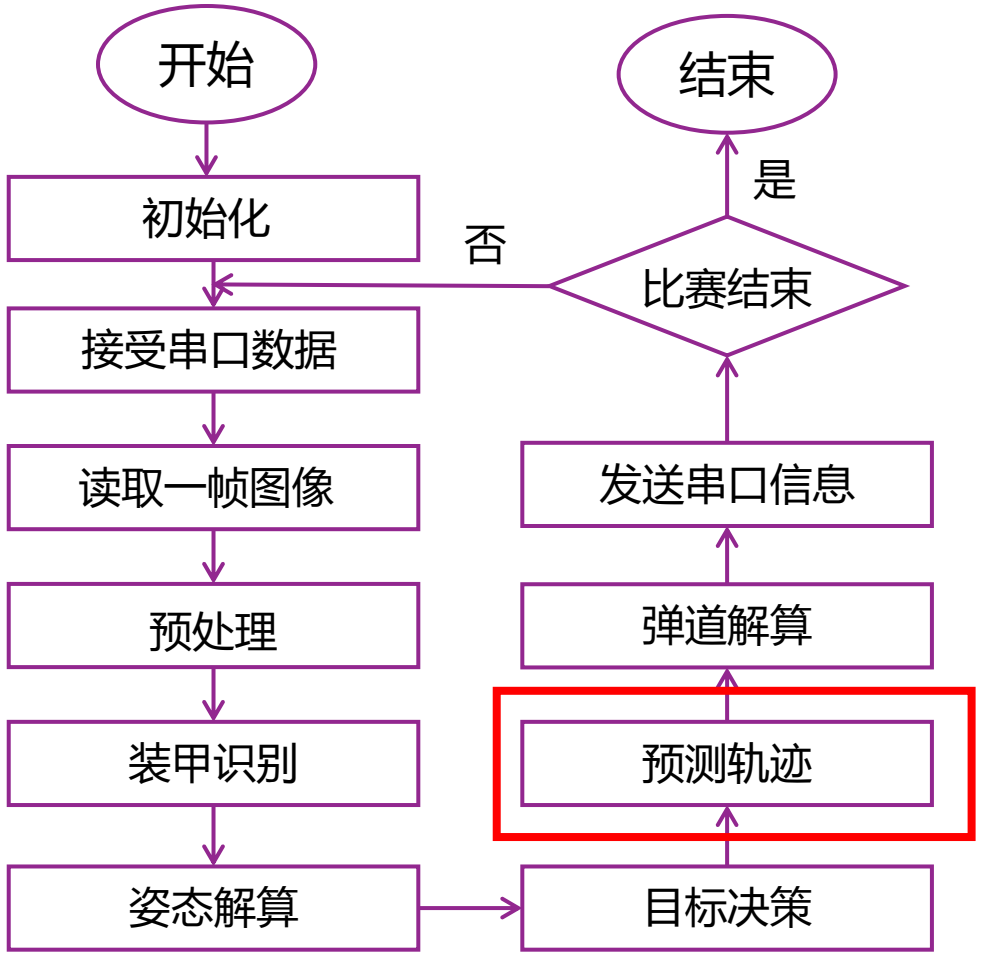
装甲板的状态转移方程

$$\begin{bmatrix} \varphi \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ & 1 \end{bmatrix} \begin{bmatrix} \varphi \\ \omega \end{bmatrix}$$

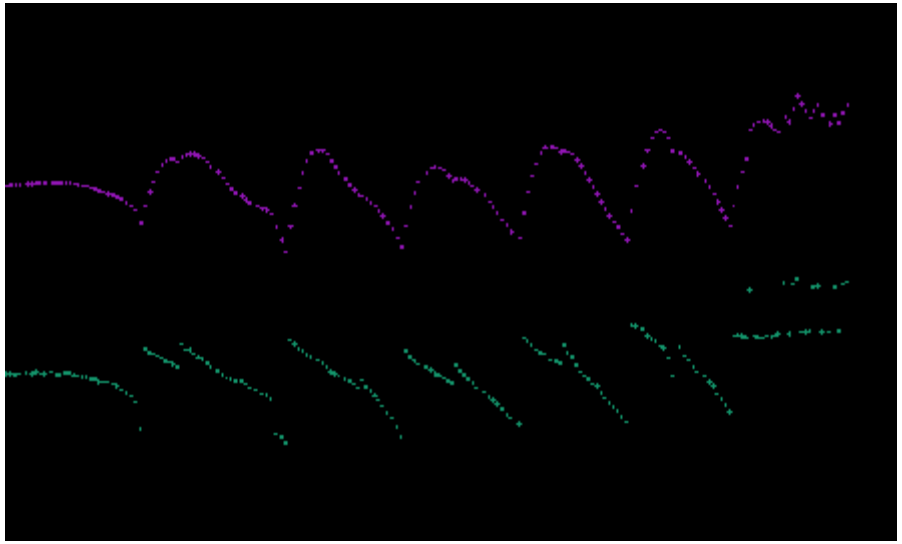
大符的状态转移方程

算法框图

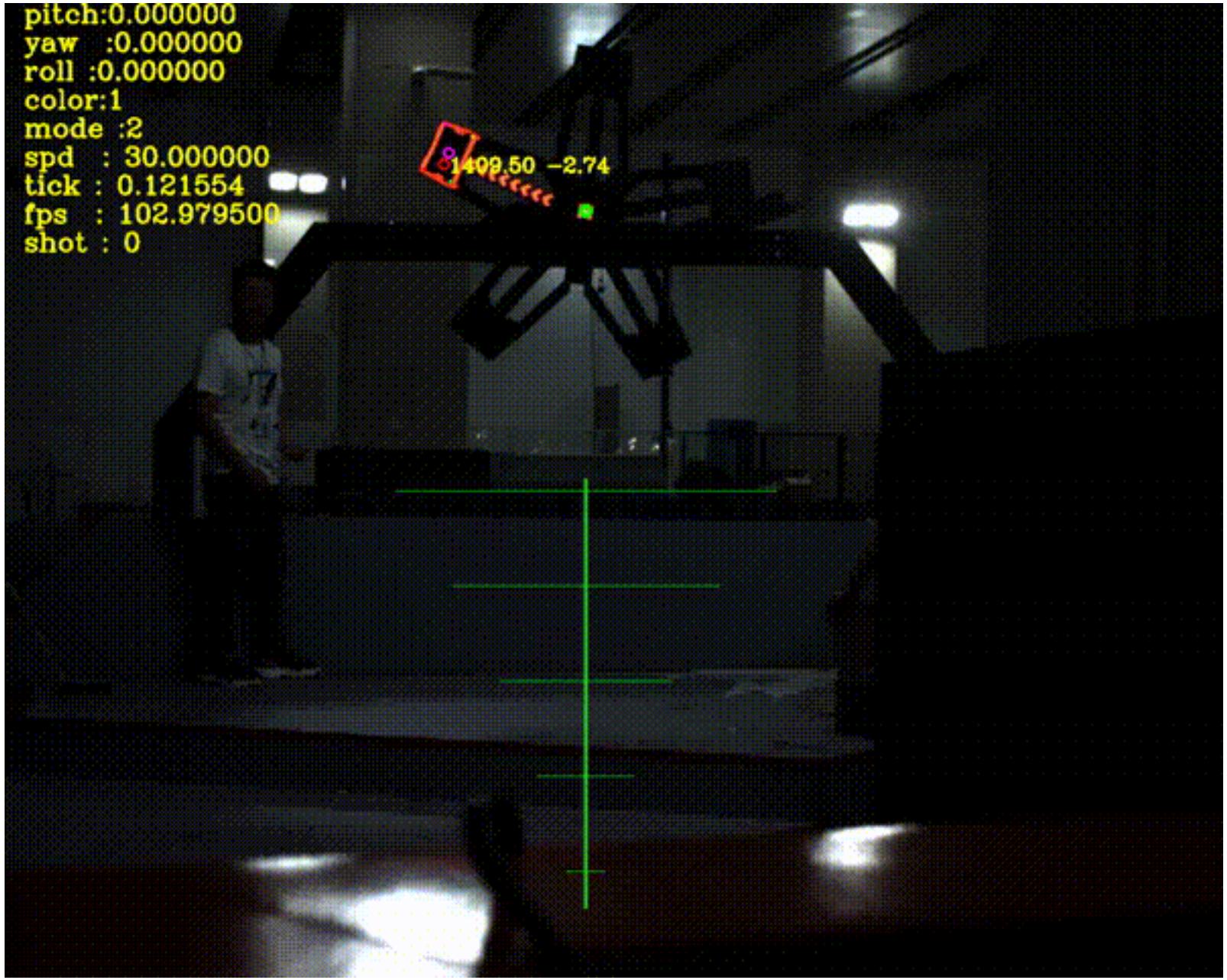

卡尔曼滤波器预测轨迹
初始化
更新状态值
读取预测值


开始
初始化
接受串口数据
读取一帧图像
预处理
装甲识别
姿态解算
目标决策
预测轨迹
弹道解算
发送串口信息
比赛结束
否
是
结束

整体流程图



```
switch (mode) {
    case NORMAL:
        measurement.at<float>(0) = (float)state.x;
        measurement.at<float>(1) = (float)state.y;
        measurement.at<float>(2) = (float)state.z;
        kf.correct(measurement);
        break;
    case C_ARM:
        kf.statePost.at<float>(0) = state.x;
        kf.statePost.at<float>(1) = state.y;
        kf.statePost.at<float>(2) = state.z;
        break;
    case C_ROBOT:
        kf.statePost.at<float>(0) = state.x;
        kf.statePost.at<float>(1) = state.y;
        kf.statePost.at<float>(2) = state.z;
        kf.statePost.at<float>(3) = 0;
        kf.statePost.at<float>(4) = 0;
        kf.statePost.at<float>(5) = 0;
        break;
```


pitch:0.000000
yaw :0.000000
roll :0.000000
color:1
mode :2
spd : 30.000000
tick : 0.121554
fps : 102.979500
shot : 0
1409.50 -2.74

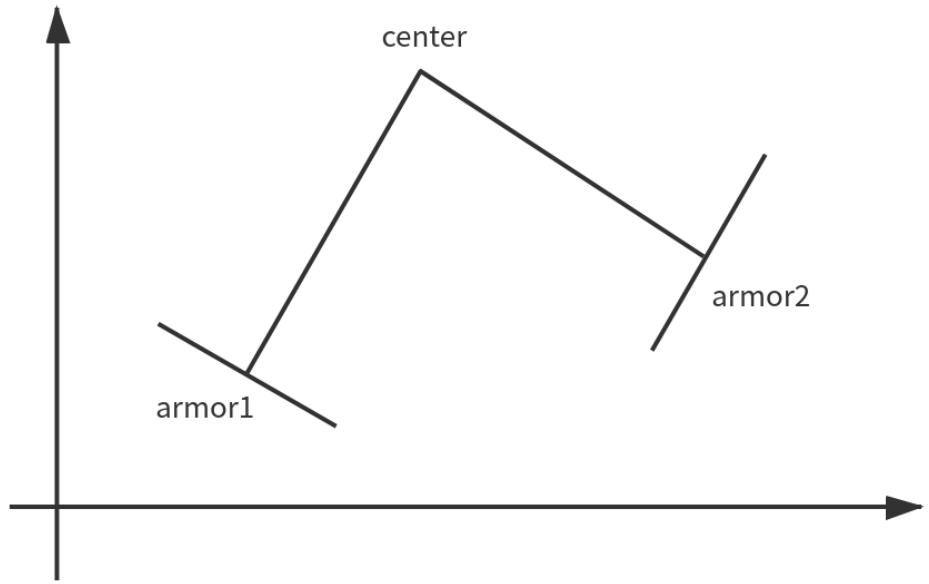
南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

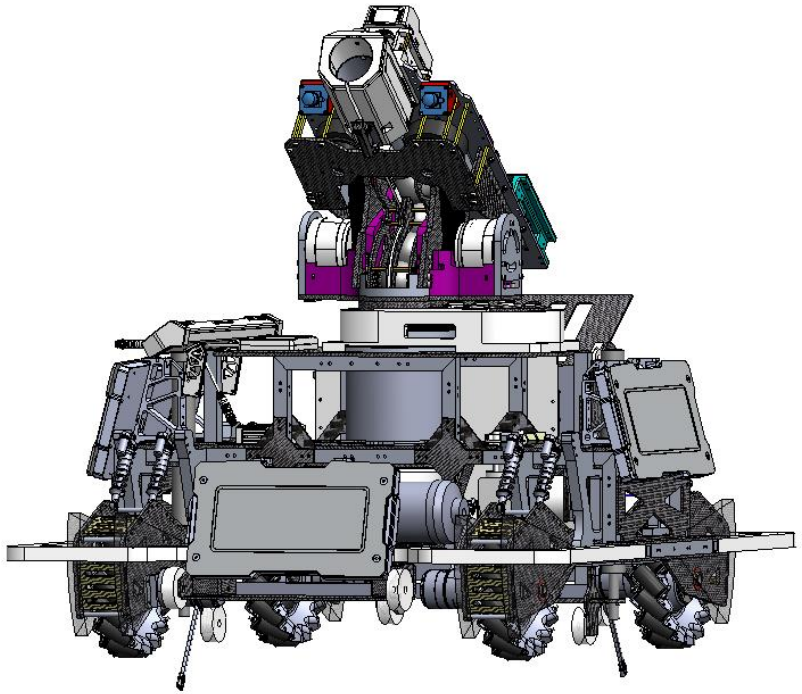
第五部分

视觉算法/PART 5

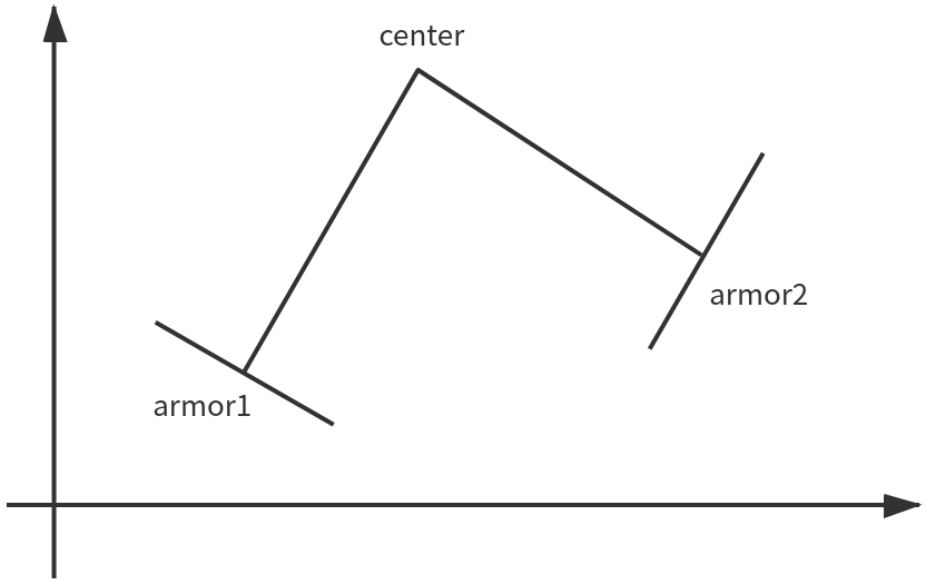
小陀螺击打

ALLIANCE





```
cv::solvePnP(NormalRealPoint, armor.concors,
             CameraMatrix, DistCoeffs,
             rvec, tvec, false, cv::SOLVEPNP_AP3P);
```



$$\begin{bmatrix} \cos(\theta_1) & -\cos(\theta_2) \\ \sin(\theta_1) & -\sin(\theta_2) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}$$

$$z = \frac{z_1 + z_2}{2}$$

目标机器人中心坐标

目标机器人角速度

旋转方向

当前装甲板角度

目标机器人中心

四块装甲板的位置

南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

感谢观看
欢迎提问