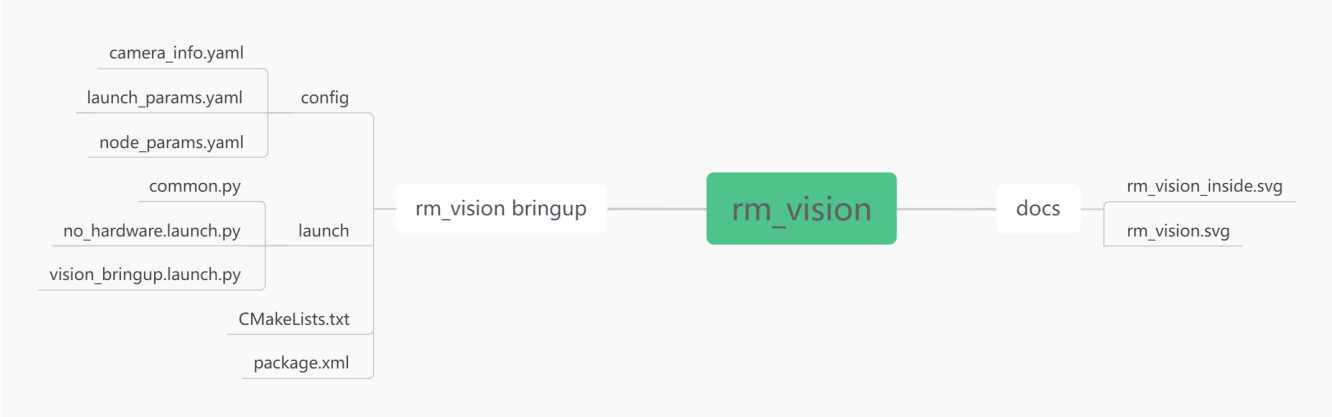
军瞄代码分析解读

rm_vision

- 文件构成目录



docs

- **rm_vision_inside.svg**

XML/HTML


```
<svg width="756" height="756" xmlns="http://www.w3.org/2000/svg" xmlns:
xlink="http://www.w3.org/1999/xlink" overflow="hidden"><defs><clipPath
id="clip0"><rect x="0" y="0" width="756" height="756"/></clipPath><line
arGradient x1="348" y1="283.5" x2="552" y2="283.5" gradientUnits="userS
paceOnUse" spreadMethod="reflect" id="fill1"><stop offset="0" stop-colo
r="#FF8B00"/><stop offset="0.5" stop-color="#FF7200"/><stop offset="1"
stop-color="#FF5900"/></linearGradient><image width="344" height="301"
xlink:href="
AB6XYgAAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAALrUExURQAAA
P//////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////wl
L9MAAAAD4dFJOUwABAgMEBQYHCAkKCwwNDg8QERITFBUWFxgZGhscHR4fICEiIyQlJicoKS
orLC0uLzAxMjM0NTY3ODk6Ozw9Pj9AQkNERUZHSElKS0xNTk9QUVJTVFZXWFlaW1xdXl9gY
WJjZGVmZ2hpamtsbW5vcHFyc3R1dnh5e3x9fn+AgYKDhIWGh4iJiouMjY6PkJGSk5SVlpeY
mZqbnJ2en6ChoqOkpaanqKmqq6ytr7CxsrO0tba3uLm6u7y+wMHCw8TFxsfIycrLzM3Oz9D
R0tPU1dbX2Nna29zd3t/g4eLj5OXm5+jp6uvs7e7v8PHy8/T19vf4+fr7/P3+wahcdwAAAA
lwSFlzAAAOwwAADsMBx2+oZAAADi9JREFUeF7t3XtAVFUeB/A7MzLAKCpoIihQVIopZoil4
gsfrairpWWkbg98lO5aW1rmIx+lLCaaqdv2Mnr5KK21LPORr8h87YorYilmICKkPBQYuH/u
nbm/e88ZuMOAMyMX/H7+ceY3B4Gvd84959xzRwEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADqLTgyOrYtPQZPaBEc2Xvw6MRFa3fvi6
cSeECreWvX7s7LEyW/j6IauM8Qbc/U7veRVAQP6IpgvQPB3qB29KczCLa+mstn+/kukWw9
dJn9OiZ0gAqI08s70clJxBsfRgy5AGUBMF6kpGikiBYT0KwXoJgvQTBegmC9RIE6yUI1ksQ
rJcowVZVVV2Po5oTCLY+jFKklZXWit1p04ZHUM2JOgTbPCi8XbAlsp0PFfekpU2+FGfekpU2Ljm7
ZwmI2majmhMtgzdHjZqdt+OrrDSuTE9sYjFS9VUmJuoxUVnuwpqgX1h757eq1kjKxquzKhW
2jOtMLehUcGdm3Fz1uWLUFawx56UB+Mb1qYy39bKAfvagzPuo10T3vUalh1RJss/vezbFW0
YvEeijRTC/ry+AlyjVR8UsqNaxagk08RC84SH+QXtaX5fTjSXQerKn3b9WOVllZ5l3UQld0
```

```
EWyz5kFtmtNjZ8EaH86hcg0LgqiNnuggWJ9WCTNeXPHRksQH7pCeOQnWOPQMVWv63zBqpCc
NHqzPgDd2ZF4sKbheXHJ4YX9/Z8EGfmWlak3WdRZqpSMNHKyh3fwfrtD3F8XSQ7OcBGt64R
IVtWTeTc105KYHG0p/ysLfL+CPxKrL4+M0g+2WWUlFTQ/Vae5xU928YANCo+LHTHw2pT09t
wlZ9wd9c8W5ZK1g/V+9TDVy9lI5PbJbpL++4CYFO3jixLkr1u8/lV/FLxuaZl2j762q5I5g
FmyndO64vrpj8VOTl66/Sk9ttupvXMCCrfJisIbM/PwqO3491jj0BH1vbSzYx7gD+8qidlK
MAd3SuH+UvfzbQB9swdqX9Ky/LqaSW5r5GOgRzyiqY3su2NabNEf8KjVY41LWw16dQiH2Ok
YVyU/hck1HUiqtv+xOmzYiunULXyrdsIDgqKgR48bGhwstqKLQvoIw+DrVnFCDbbOFKpLdr
anYfDFVJMe7UVE/BnSv0yqpaz4hgxamHc7Jysg6dSx10v2Oa06awTbj+ndNarBRv1BFmsA+
TzVBmEQlyYl7qaYfJvcjlU34YN/FMvup2moV/0h7xeGg1QzWsodKHIe+QQ32/lyqiGJOT6o
JQgLrH3QYrGeYYj/Id1jQq8h+hz+faAVrCOU6SVEsObFucdqhs/TMTg12MJsdnIyimvRmK6
Fa0w227ROn6Ddkcjb4sdOYVrDGvueoZHPls8SIoMCgGQfpuY0SrCmxgCqiuJOdpvoVUa1RB
2vwbRUeEmBoqXWNqc2yXH5ZX7G0A72uHaxp+EUqSaxLA+xFv8czqCJRgjVPZ2PWz9kboUkE
a7p92MyPP/n0rZlDWVqK1itzNIdNl5+kBk6CfZTFVXkkmqohq6kkUYL1/Rv7d9vYpII19Vq
9P6ukrLxMLNq97r5qZztL8lUn8/if1P5QM9i/sD7SOkeZk5qmlFGNBes/r2kFK83t4xNtd1
12mHecRXftUE/HAe+I8/RKDdalypV/zWCnsOl+/iAqCoZ+7DhWgrUsYv8EjTvYjvblEtvc/
mSs9GxNnsNa6E6pxsQddb5QejCMGmkHy74w83YqCkL3fKpxwS4spYoobgi2VQwGg7HLs2xO
20iCjVj37/00t8/rK4S+W+jYhZa/1ZIa2mx0nquYM5gaaQb7DPvKo/Kpy0ZjPZYPdmN7Q7t
7BkyZs+KL7cfZ1+s32NCoXkMG0GMhNtu+XGJzqW/rBdVW7KTjayAbG9z/GxVtCjPOcmd6aZ
o0j1ppBvscVSQH2MjMRbA/v73iu33HsoodhyE6DNang/K2P/ABlaRg6eeVgu0zSeMa3nL1A
At4nztg81InDnliVSFX2UqBuQp2L9UkLoItZg85Ogw2YeV65W2vLhtyweZN3UWPeIfUywAx
F9RuwvpxuJ/R5GMaup/1HOkeD1abDoNdbttaKSehFWzJIa2hf0FfaiksVE8g1jV3Um0aO1u
fu5WDVWkFq630KWUs+z1VRDG3t7IzoLt9flt+vehKYRaCldQ92LLZFOztWVQRxeRWckk6EW
4rKiw88uXqx/pHKCc5rwfbQ26pIzcUrHVxM7llPEthpDoh8x+QGBFhNBqlwSZVvBzshfQ3a
060G1qdgj1x8Eg+P5atTKUp1Tg2TeqhpmgbvtMjhbeCTd8yZ86wYZ3uaO+hRWUPch1sxcGV
I4fGjT9KT20qV9HWySR2ccVhOladx4O9IEX6pJRox8Dm+tzF6TrYPxbaz/YBsdzeqco36be
Zwab7j9S2ad1zwZYftSXaTUo00KzTSGWugrV+EiL//Oa57BK0Gux0FuyHt8klTZ4LdnuMfg
9SB66CvTBSORIHss5ADfZp1hWcnRBUvWdlPBfsZ/ZFmEbAVbC71HWniK+oxAU7jttIcXrJ+
FAfQbtD8Fyw3LKhvrkI1vqmupLVMo1qXLC9+d644PjO+bExkVrbExCsHRfs1b+rGwRM/6Qa
F2xwOlVkpaX5B1JfHxUmVN+j5s1g20bFJc1oQ0/0w0WwhY/TVEA7WGFZjc0s1stZO18eHeN
PDWTeCbZzv6SXUlK+2bv39IFIKumHi2AL1HOXdrDx/HIsqbRac96e5bAX2M1gF7Fg5SsIdl
2+3ns6R17w+dHFfaQNwEWwl9TVb+1gw9fxMzKm4tzn/LqIm8HOZ8FuZhuXp1BJotPdhkQzW
HWBUDtYc/QZJ5dmrBkJ6rKMm8H6v8guJu5QFieFoHeoJPlCz/tjbyhYQZh9koo1nBxOTdwN
1vc5tih8Sn0jxLGVNXG+/u77dDvYVtNOVlC5msp991AbN4M1P8mCzZ0g1wThbW7jMTXUE7e
DFQL+ukW7nxWrkpUxrctg1UmboUfNYE0JbBHt+ndyd+ozrJAqkjON5K6Z+gUrWOJTL5Robo
Y5qmwcchVsujqrMP2J7SxUghX6sLBFMS2hU7dBY9/j7wpZRe30xAPBCibf4clbT7ETjKp4O
rVwFexx9axuerbmThghmu/Hywq2b9h/mrUSxcwh1E5PPBGsxK/7kOc/ys24VK273UFvcVfB
nlZ7Y8s7bGilBhvKbZXXsL4jtdMTDwUrdXqBrTqMmZtyxKG/LaUrDa6CzR1DRaGHxjZOwfi
qk213dnm6/PB5jwVrZwrtO/sM3ydEy4esZrAzqSK5voBu2fBPZpsNWbDCJO5MVV3RMzq8k9
bTwUqtmj3wIZfN2FqC5fZuWffK2w1vm8JvXWTBRv9AJQ2b2YY6PfF0sJKBmdRMMr2WYKdyc
7aiTV2lrrTX8hz+Tk4WbOBcrX0jNtZtd9f2szQcLwQbuIb1s7NrCXYyd2SL104tmbtqFz+s
```

```
4oM1dnxPe+ZccXCQ/i7Q2nkhWGEMGxu8VEuwjzochVU1528sWMF472EqOri8ZZQ+j1fvBBv
LDsUZzoM1clMBbVywQsCfr7AtYaq1fXR6vHon2N6so3yolmDjnO6xJ3ywgmH4FnYbnU1Vcb
o+xwMyTwTbNipuMv//7Uxg/eEDzoM1dK55c5iE60sdghVMXVMPSJ2q3H9fyz26eWIn+S/X
J/eCvTuOLo5kfcs2u5uWqSevcvrfJLSCFfy1tt5aM1myjsEKQtCQqbv+c+p0Vk72r5vmJ0Q
E6LYbsHEr2MCP1Isjl8PVw6cHu5Uzgy7saAbrk0Il3pFZ7C7E6sEKRt+QHg+OfzS2Z/s2Fp
OeD1eJW8EGH2MDqynK5cNWG9gFxiX022sGKwypeTq69ngXNuTKGUENHZh8PXVftVe5d8RuY
3P4/fJ2Q9/IRdmsJ1AOOe1gQ76gmqp4mcW+0F1pLS8vK/uxPzVsjNwKNoAbthevjQ9paWox
ciO3NPuNsp9LO1if0dWu8Ra94id0zS0vu/bz9jfeSErqqcdPgKsrt4L1m8/Ph7b+69MV32d
zlbKRSj+oHaxgeYYfypYffTpMEO6al5Q0NC4mrK2Oh1J14VawwmPsw8ikE3pxueOM/gf1uo
CTYIWWM8+K8qC3oihrdf9AqWRub2nkkcrcCzasto8huqjcl+g8WCEwZvO3h4+fOZ+9f/3UM
MfdM42ce8GakvnVqGrmUCOJ02Cls13UIwnxPTu01feotP7cC1bomcUGXNX8yL2jlWCrKq0l
NT6u32DyaWKh2rgZrPk1jWuIdt/zH5drlBKVBlC2s31KJ6o1cW4GK0Ru1Fh0kk5ke9iH4ki
M56VIX04aLZ3tg5rEqck1d4MVEr7W6AwK3r3H8d3dLyYmrGmc7evK7WAt475jF6xlxVnTb2
MNblFuByuYey/nr7GI4vlPHlZuq72FuR+sYLYkffnf322HbWnRpV92vT9If/vWG4AHgpUGT
H5jFixO+XbTxjWT4+9U99bf2jwSrMSvfUBoy6BAi7kJDklvSP2DlYakFam3/LnJpfoEW2Vf
Jc3+efuKYQjWlXoE+9ox2yA/KS4mKsyCN7wrtQdbxQVr6Nyv8a+S3jzOgpUvjmTvVDeuQv1
oB2tfLrG97aMx1L9BWsF2fKRJXBxpWLZgK61W6W1/IZVKgtmMSN32jwrlmmj/LlQCTxg7B2
97rwhqjkgBAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABqKIPwfZYxNPlJPWvIAAAAASUVORK5CYII
=" preserveAspectRatio="none" id="img2"></image></defs><g clip-path="ur
l(#clip0)"><rect x="0" y="0" width="756" height="756"/><path d="M172 13
2.934C172 113.088 188.088 97 207.934 97L548.066 97C567.912 97 584 113.0
88 584 132.934L584 624.066C584 643.912 567.912 660 548.066 660L207.934
660C188.088 660 172 643.912 172 624.066Z" fill="#5A5655" fill-rule="eve
nodd"/><path d="M207.935 96 548.065 96C567.912 96 584 112.112 584 131.9
88L584 379.534 573.958 387.055C518.021 424.901 450.587 447 378 447 305.
413 447 237.979 424.901 182.042 387.055L172 379.534 172 131.988C172 11
2.112 188.089 96 207.935 96Z" fill="#FFFFFF" fill-rule="evenodd"/><pat
h d="M241 167 400.937 167.143C400.937 167.143 446.8 173.217 447.979 22
2.138 449.157 271.059 400.166 277.951 400.166 277.951L317.61 278.682 29
9.653 344 249.139 343.771 277.878 238.98 396.762 238.513C396.762 238.51
3 407.391 236.37 407.289 222.082 407.187 207.794 393.594 206.672 393.59
4 206.672L334.826 206.569 358.742 233.921 301.764 234.084 241 167Z" fil
l="#5A5655" fill-rule="evenodd"/><path d="M348 282.85 405.207 282.824 4
19.981 299.637 496.036 200.061 552 200 425.749 367" fill="url(#fill1)"
fill-rule="evenodd"/><use width="100%" height="100%" xlink:href="#img
2" fill="none" transform="translate(206 420)"></use></g></svg>
```


- **rm_vision.svg**

XML/HTML

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<!-- Created with Vectornator (http://vectornator.io/) -->
<svg height="100%" stroke-miterlimit="10" style="fill-rule:nonzero;clip-rule:evenodd;stroke-linecap:round;stroke-linejoin:round;" version="1.1" viewBox="0 0 1024 1024" width="100%" xml:space="preserve" xmlns="http://www.w3.org/2000/svg" xmlns:vectornator="http://vectornator.io" xmlns:xlink="http://www.w3.org/1999/xlink">
<defs>
<linearGradient gradientTransform="matrix(1 0 0 1 6.39488e-14 157.46)" gradientUnits="userSpaceOnUse" id="LinearGradient" x1="375.972" x2="944.796" y1="453.158" y2="453.158">
<stop offset="0" stop-color="#ff8b00"/>
<stop offset="1" stop-color="#ff5900"/>
</linearGradient>
</defs>
<g id="图层-2" vectornator:layerName="图层 2">
<path d="M79.2038 285.449L524.109 285.848C524.109 285.848 651.69 302.769 654.969 439.05C658.247 575.332 521.965 594.533 521.965 594.533L292.315 596.568L242.362 778.529L101.844 777.892L181.788 485.967L512.496 484.668C512.496 484.668 542.064 478.698 541.78 438.895C541.496 399.092 503.683 395.965 503.683 395.965L340.206 395.68L406.734 471.875L248.235 472.328L79.2038 285.449Z" fill="#5a5655" fill-rule="nonzero" opacity="1" stroke="none"/>
</g>
<g id="图层-3" vectornator:layerName="图层 3">
<path d="M375.972 608.813L535.485 608.74L576.68 655.417L788.749 378.975L944.796 378.807L592.763 842.427" fill="url(#LinearGradient)" fill-rule="nonzero" opacity="1" stroke="#5a5655" stroke-linecap="butt" stroke-linejoin="miter" stroke-width="0"/>
</g>
</svg>

```

- **svg 文件的一般用处？**

SVG 文件在网页开发中有多种常见用途。首先，SVG 文件是矢量图形文件，可以在不失真的情况下进行缩放，适用于响应式设计。其主要用途包括但不限于：

i. **图标和标识**：SVG 图标可用于网站的导航菜单、按钮或其他界面元素，因为它们可以在不同大小的屏幕上保持清晰度。
ii. **数据可视化**：SVG 可用于创建图表、图形和数据可视化，使数据更易于理解和呈现。
iii. **动画效果**：通过 CSS 和 JavaScript，可以为 SVG 图形添加动画效果，例如平滑过渡、交互效果和形态变换。
iv. **背景图案**：SVG 可用于创建复杂的背景图案或纹理，为网页增添视觉吸引力。
v. **网页装饰**：SVG 可用于装饰网页元素，如分隔线、装饰性图案或其他视觉元素。

总的来说，SVG 文件在网页开发中被广泛应用，因为它们具有响应式设计、清晰度高、易于动画化等优点，适合用于各种视觉和图形需求。

∘ 这段 svg 代码的含义？

这段代码创建了一个可缩放矢量图形，用于网页设计或数据可视化等用途，它描述了一个包含裁剪路径、渐变填充、图像插入和形状绘制的图形元素组合。通过这种元素的组合和设置，可以保持图形的清晰度和可缩放性。

i. `<svg>` 标签定义了一个 SVG 图形，指定了宽度和高度，并设置了命名空间。
ii. `<defs>` 标签用于定义 SVG 图形中的元素，如裁剪路径和渐变。
iii. `<clipPath>` 定义了一个裁剪路径，用于限制图形的可见区域。
iv. `<linearGradient>` 定义了一个线性渐变，用于填充图形的一部分。
v. `<image>` 标签插入了一个图像，使用了 Base64 编码的 PNG 图像数据作为图像源。
vi. `<g>` 标签用于将多个图形元素组合在一起，并应用了裁剪路径。
vii. `<rect>` 和 `<path>` 元素描述了矩形和路径形状，填充颜色为指定的颜色。
viii. `<use>` 元素用于重复使用另一个图形元素，并设置了填充颜色和变换。

∘ svg 文件对机甲大师赛事的视觉识别系统中的用处一般应用？

i. **显示比赛场景**：SVG 图形可以用来呈现比赛场景的地图、机器人位置、障碍物等元素，帮助观众和裁判了解比赛情况。
ii. **标识机器人位置**：通过 SVG 中的图形元素，可以标识和显示机器人在比赛场景中的位置和移动轨迹，帮助观众和裁判跟踪机器人的运动。
iii. **视觉辅助**：SVG 图形可以用于创建视觉辅助元素，如标记区域、路径规划、目标点等，帮助机器人在比赛中执行任务和导航。
iv. **数据可视化**：SVG 图形可以用于可视化比赛数据，如得分情况、比赛进展等，使信息更直观和易于理解。

rm_vision_bringup

config

yaml 代码的用处？

YAML 文件通常用于存储配置信息、数据序列化和数据交换。它是一种人类可读的数据序列化语言，常用于配置文件、数据传输和存储。YAML 文件的主要用途包括但不限于：配置文件、数据交换、元数据存储、API 请求和响应等。

yaml 代码在机甲大师赛事中的场景用处？

1. **配置机器人行为**：使用 YAML 文件定义机器人的运动、射击、避障等行为逻辑，以实现自动化控制和决策。
2. **定义比赛规则和场景**：使用 YAML 文件描述比赛规则、场地布局、目标位置等信息，帮助机器人理解比赛环境。
3. **参数调整和优化**：通过 YAML 文件调整机器人的参数，如速度、转向角度、射击精度等，以优化机器人的表现和策略。
4. **数据交换和通信**：使用 YAML 格式进行机器人之间的数据交换和通信，传输状态信息、目标位置等数据。

总的来说，YAML 代码在机甲大师比赛中可以帮助团队配置机器人行为、优化策略，并促进机器人之间的数据交换和通信，从而提升比赛表现和竞争力。

camera_info.yaml

YAML

```yaml
image_width: 1440
image_height: 1080
camera_name: narrow_stereo
camera_matrix:
  rows: 3
  cols: 3
  data: [1807.12121,    0.      ,  711.11997,
            0.      , 1806.46896,  562.49495,
            0.      ,    0.      ,    1.      ]
distortion_model: plumb_bob
distortion_coefficients:
  rows: 1
  cols: 5
  data: [-0.078049, 0.158627, 0.000304, -0.000566, 0.000000]
rectification_matrix:
  rows: 3
  cols: 3
  data: [1., 0., 0.,
         0., 1., 0.,
         0., 0., 1.]
projection_matrix:
  rows: 3
  cols: 4
  data: [1790.52356,    0.      ,  710.04861,    0.      ,
            0.      , 1794.3208 ,  562.32704,    0.      ,
            0.      ,    0.      ,    1.      ,    0.      ]

```

这段代码描述了对于相机的参数和校准信息，具体分布解读如下：

- `image_width: 1440` 和 `image_height: 1080` 指定了图像的宽度和高度。
- `camera_name: narrow_stereo` 指定了相机的名称为 narrow_stereo。
- `camera_matrix` 包含了相机的内参矩阵，其中 `rows: 3` 和 `cols: 3` 表示矩阵的行数和列数，`data` 包含了内参矩阵的具体数值。
- `distortion_model: plumb_bob` 指定了畸变模型为 plumb_bob。
- `distortion_coefficients` 包含了畸变系数，`rows: 1` 和 `cols: 5` 表示系数的行数和列数，`data` 包含了畸变系数的具体数值。

- `rectification_matrix` 包含了矫正矩阵，用于校正图像。
- `projection_matrix` 包含了投影矩阵，用于将相机坐标系中的点映射到图像平面上。

这段代码描述了对于相机的参数信息，包括内参矩阵、畸变系数、矫正举证和投影矩阵，这些信息对于相机的矫准和 图像处理非常的重要，帮助我们把控相机在获取装甲板的图像信息时的校准距离问题，越正确的参数越能提高在自瞄时的容错率。

launch_params.yaml

```yaml
camera: hik

odom2camera:
  xyz: "\"0.10 0.0  0.05\""
  rpy: "\"0.0  0.0  0.0\""

detector_log_level: INFO
tracker_log_level: INFO
serial_log_level: INFO
```

这段代码描述了相机和机器人的关系以及一些日志级别设置，具体解读如下：

1. `camera: hik` 指定了相机的类型为"海康"。
2. `odom2camera` 部分描述了机器人坐标系到相机坐标系的转换关系：
   - `xyz: "\"0.10 0.0 0.05\""` 表示机器人坐标系原点在相机坐标系中的位置为(x=0.10, y=0.0, z=0.05)。
   - `rpy: "\"0.0 0.0 0.0\""` 表示机器人坐标系到相机坐标系的旋转角度为 roll=0.0, pitch=0.0, yaw=0.0。
3. `detector_log_level: INFO` 设置了检测器的日志级别为 INFO，表示输出信息级别，通常用于记录程序运行状态和关键信息。
4. `tracker_log_level: INFO` 设置了跟踪器的日志级别为 INFO，用于记录跟踪器的运行状态和信息。
5. `serial_log_level: INFO` 设置了串口的日志级别为 INFO，用于记录串口通信的状态和信息。

这段代码主要用于配置相机和机器人之间的关系，以及设置不同模块的日志级别，帮助调试和监控系统运行状态。

node_params.yaml

YAML

```yaml
/camera_node:
  ros__parameters:
    camera_info_url: package://rm_vision_bringup/config/camera_info.yaml
    exposure_time: 2500
    gain: 8.0

/serial_driver:
  ros__parameters:
    timestamp_offset: 0.006
    device_name: /dev/ttyACM0
    baud_rate: 115200
    flow_control: none
    parity: none
    stop_bits: "1"

/armor_detector:
  ros__parameters:
    debug: true

    detect_color: 0
    binary_thres: 80

    light.min_ratio: 0.1
    armor.min_light_ratio: 0.8

    classifier_threshold: 0.8
    ignore_classes: ["negative"]

/armor_tracker:
  ros__parameters:
    target_frame: odom
    max_armor_distance: 10.0

    ekf:
      sigma2_q_xyz: 0.05
      sigma2_q_yaw: 5.0
      sigma2_q_r: 80.0
```

```
38
39        r_xyz_factor: 4e-4
40        r_yaw: 5e-3
41
42      tracker:
43        max_match_distance: 0.5
44        max_match_yaw_diff: 1.0
45
46        tracking_thres: 5
47        lost_time_thres: 1.0
48
```

这段代码是一个 ROS 参数配置文件，用于配置不同节点的参数。(新代码的涉及大量的 ros 部分，需要尽快掌握)以下是对每个节点的参数配置的解读：

1. `/camera_node` 节点：
   - `camera_info_url: package://rm_vision_bringup/config/camera_info.yaml` 指定了相机信息的 URL 路径。
   - `exposure_time: 2500` 设置了曝光时间为 2500。
   - `gain: 8.0` 设置了增益为 8.0。
2. `/serial_driver` 节点：
   - `timestamp_offset: 0.006` 设置了时间戳偏移为 0.006。
   - `device_name: /dev/ttyACM0` 指定了设备名称为 `/dev/ttyACM0` 。
   - `baud_rate: 115200` 设置了波特率为 115200。
   - `flow_control: none` 设置了流控为无。
   - `parity: none` 设置了奇偶校验为无。
   - `stop_bits: "1"` 设置了停止位为 1。
3. `/armor_detector` 节点：
   - `debug: true` 启用了调试模式。
   - `detect_color: 0` 指定了检测颜色为 0。
   - `binary_thres: 80` 设置了二值化阈值为 80。
   - `light.min_ratio: 0.1` 设置了最小灯条比例为 0.1。
   - `armor.min_light_ratio: 0.8` 设置了最小装甲板灯条比例为 0.8。
   - `classifier_threshold: 0.8` 设置了分类器阈值为 0.8。
   - `ignore_classes: ["negative"]` 指定了忽略的类别为"negative"。
4. `/armor_tracker` 节点：
   - `target_frame: odom` 设置了目标坐标系为 odom。
   - `max_armor_distance: 10.0` 设置了最大装甲板距离为 10.0。

- `ekf` 部分包含了扩展卡尔曼滤波器的参数配置。
- `tracker` 部分包含了追踪器的参数配置。

这段代码用于配置不同节点的参数，包括相机节点、串口驱动节点、装甲检测节点和装甲追踪节点，以便在系统中正确设置和调整各个节点的功能和行为。

launch

common.py

Python

```python
import os
import yaml

from ament_index_python.packages import get_package_share_directory
from launch.substitutions import Command
from launch_ros.actions import Node

launch_params = yaml.safe_load(open(os.path.join(
    get_package_share_directory('rm_vision_bringup'), 'config', 'launch_params.yaml')))

robot_description = Command(['xacro ', os.path.join(
    get_package_share_directory('rm_gimbal_description'), 'urdf', 'rm_gimbal.urdf.xacro'),
    ' xyz:=', launch_params['odom2camera']['xyz'], ' rpy:=', launch_params['odom2camera']['rpy']])

robot_state_publisher = Node(
    package='robot_state_publisher',
    executable='robot_state_publisher',
    parameters=[{'robot_description': robot_description,
                 'publish_frequency': 1000.0}]
)

node_params = os.path.join(
    get_package_share_directory('rm_vision_bringup'), 'config', 'node_params.yaml')

tracker_node = Node(
    package='armor_tracker',
    executable='armor_tracker_node',
    output='both',
    emulate_tty=True,
    parameters=[node_params],
    ros_arguments=['--log-level', 'armor_tracker:='+launch_params['tracker_log_level']],
```

```
32 )
33
```

这段代码是一个ROS2启动文件，用于配置和启动不同的节点。以下是对代码的解读：

a. 首先，代码通过导入必要的库和模块，准备加载配置文件和设置节点参数。
b. 使用`get_package_share_directory`函数获取ROS包的共享目录路径，以便加载配置文件和URDF文件。
c. 通过`yaml.safe_load`函数加载`launch_params.yaml`配置文件，其中包含了启动节点所需的参数信息，如`odom2camera`参数。
d. 创建`robot_description`，其中包含了URDF文件路径和`odom2camera`参数，用于描述机器人的外观和状态。
e. 使用`Node`类创建`robot_state_publisher`节点，该节点用于发布机器人的状态信息。在节点参数中，包括了机器人描述和发布频率等参数。
f. 加载`node_params.yaml`配置文件，该文件包含了节点的参数信息。
g. 使用`Node`类创建`tracker_node`节点，该节点用于执行`armor_tracker_node`可执行文件。设置了输出方式、日志级别等参数，通过`ros_arguments`传递日志级别参数。

总体来说，这段代码的目的是配置和启动两个ROS 2节点，其中`robot_state_publisher`节点用于发布机器人状态信息，而`tracker_node`节点用于执行特定的节点功能。通过加载配置文件和设置参数，实现了节点的启动和功能配置。

- **no_hardware.launch.py**

Python

```python
import os
import sys
from ament_index_python.packages import get_package_share_directory
sys.path.append(os.path.join(get_package_share_directory('rm_vision_bringup'), 'launch'))


def generate_launch_description():

    from common import launch_params, robot_state_publisher, node_params, tracker_node
    from launch_ros.actions import Node
    from launch import LaunchDescription

    detector_node = Node(
        package='armor_detector',
        executable='armor_detector_node',
        emulate_tty=True,
        output='both',
        parameters=[node_params],
        arguments=['--ros-args', '--log-level',
                   'armor_detector:='+launch_params['detector_log_level']],
    )

    return LaunchDescription([
        robot_state_publisher,
        detector_node,
        tracker_node,
    ])

```

`no_hardware.launch.py` 这个文件名暗示了一个 ROS 启动文件（launch file）的 Python 脚本，可能用于在没有硬件连接的情况下进行仿真或测试。在 ROS 中，通常使用 .launch 文件扩展名来表示启动文件，而 .py 文件扩展名表示 Python 脚本。因此，`no_hardware.launch.py` 可能包含了一些节点配置和参数设置，用于模拟或测试某些功能，而无需实际硬件设备。代码分析如下：

a. 首先，通过导入必要的库和模块，包括`os`、`sys`、`ament_index_python.packages`，以及从`get_package_share_directory`获取包共享目录路径，并将路径添加到Python模块搜索路径中。

b. 定义了一个函数`generate_launch_description()`，用于生成启动描述。在函数中：

a. 从`common`模块中导入`launch_params`、`robot_state_publisher`、`node_params`和`tracker_node`，这些是用于配置和启动节点的参数和节点信息。

b. 创建了一个`detector_node`节点，该节点用于执行`armor_detector_node`可执行文件，设置了输出方式、日志级别等参数，并传递了一些参数作为节点的启动参数。

c. 返回了一个`LaunchDescription`对象，其中包含了`robot_state_publisher`、`detector_node`和`tracker_node`节点，这些节点将在启动时一起启动。

总体来说，这段代码的作用是生成一个包含机器人状态发布节点、检测器节点和跟踪器节点的ROS 2启动描述，以便在启动时同时启动这些节点。

- ## vision_bringup.launch.py

Python

```python
import os
import sys
from ament_index_python.packages import get_package_share_directory
sys.path.append(os.path.join(get_package_share_directory('rm_vision_bringup'), 'launch'))


def generate_launch_description():

    from common import node_params, launch_params, robot_state_publisher, tracker_node
    from launch_ros.descriptions import ComposableNode
    from launch_ros.actions import ComposableNodeContainer, Node
    from launch.actions import TimerAction, Shutdown
    from launch import LaunchDescription

    def get_camera_node(package, plugin):
        return ComposableNode(
            package=package,
            plugin=plugin,
            name='camera_node',
            parameters=[node_params],
            extra_arguments=[{'use_intra_process_comms': True}]
        )

    def get_camera_detector_container(camera_node):
        return ComposableNodeContainer(
            name='camera_detector_container',
            namespace='',
            package='rclcpp_components',
            executable='component_container',
            composable_node_descriptions=[
                camera_node,
                ComposableNode(
                    package='armor_detector',
                    plugin='rm_auto_aim::ArmorDetectorNode',
                    name='armor_detector',
                    parameters=[node_params],
```

```
37                      extra_arguments=[{'use_intra_process_comms': True}]
38                  )
39              ],
40              output='both',
41              emulate_tty=True,
42              ros_arguments=['--ros-args', '--log-level',
43                             'armor_detector:='+launch_params['detector_l
   og_level']],
44              on_exit=Shutdown(),
45          )
46
47      hik_camera_node = get_camera_node('hik_camera', 'hik_camera::HikCam
   eraNode')
48      mv_camera_node = get_camera_node('mindvision_camera', 'mindvision_c
   amera::MVCameraNode')
49
50      if (launch_params['camera'] == 'hik'):
51          cam_detector = get_camera_detector_container(hik_camera_node)
52      elif (launch_params['camera'] == 'mv'):
53          cam_detector = get_camera_detector_container(mv_camera_node)
54
55      serial_driver_node = Node(
56          package='rm_serial_driver',
57          executable='rm_serial_driver_node',
58          name='serial_driver',
59          output='both',
60          emulate_tty=True,
61          parameters=[node_params],
62          on_exit=Shutdown(),
63          ros_arguments=['--ros-args', '--log-level',
64                         'serial_driver:='+launch_params['serial_log_leve
   l']],
65      )
66
67      delay_serial_node = TimerAction(
68          period=1.5,
69          actions=[serial_driver_node],
70      )
71
72      delay_tracker_node = TimerAction(
```

```
        period=2.0,
        actions=[tracker_node],
    )

    return LaunchDescription([
        robot_state_publisher,
        cam_detector,
        delay_serial_node,
        delay_tracker_node,
    ])

```

`vision_bringup.launch.py` 这个文件名暗示了一个 ROS 2 启动文件（launch file）的 Python 脚本，可能用于配置和启动与视觉相关的节点或功能。在 ROS 中，通常使用 .launch 文件扩展名来表示启动文件，而 .py 文件扩展名表示 Python 脚本。因此，`vision_bringup.launch.py` 可能包含了一些节点配置和参数设置，用于启动与视觉功能相关的节点或模块。