

机器人学习中的相机标定教程

Camera Calibration for Robot Learning ·完整中文版

2026 年 10 月 15 日

目录

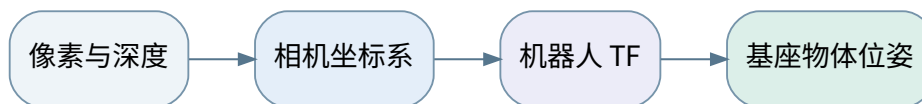
- 一、相机标定的分类与用途 2
- 二、最重要的数学概念与坐标系 2
- 三、相机内参标定 (Intrinsic Calibration) 3
- 四、相机外参标定 (Extrinsic Calibration) 4
- 五、手眼标定 (Hand-Eye Calibration) 5
- 六、Tron2 头部与腕部相机的特殊情况 6
- 七、RGB-D 深度相机标定与对齐 7
- 八、OpenCV 完整手眼标定代码 8
- 九、ROS 2 中的标定工具与基本操作 10
- 十、将标定结果写入 ROS TF2 / URDF 10
- 十一、相机标定如何进入机器人学习 11
- 十二、标定精度与结果验证 13
- 十三、常见问题与排查 14
- 十四、推荐开源工具与官方资料 14
- 十五、针对 Tron2 的实施路线 14
- 十六、总结与检查清单 15

课程首页 ·English ·PDF

项目	说明
适用范围	OpenCV 4.x 概念; ROS 2 Jazzy; 设备 SDK 版本需另行匹配
文档核验日期	2026-10-18
许可	CC BY-NC-SA 4.0 ·Xiong Zeng and contributors

一、相机标定的分类与用途

章节概念图



在机器人学习中，“相机标定”不仅指计算相机内参，还包括建立相机、机器人和环境之间的空间及时序关系。

标定类型	英文名称	主要目的	常见应用
相机内参标定	Intrinsic Calibration	确定成像几何与畸变	2D-3D 投影、PnP、6D Pose
相机外参标定	Extrinsic Calibration	确定相机相对参考系的位姿	机器人抓取、空间定位
手眼标定	Hand-Eye Calibration	确定相机与末端/基座之间的刚体变换	腕部/固定相机标定
多相机标定	Multi-Camera Calibration	确定多相机之间的相对位姿	视觉融合、多视角感知
深度相机标定	RGB-D / Depth Calibration	校验 RGB-Depth 对齐和深度尺度	点云、三维定位
时间标定	Temporal Calibration	对齐图像、关节状态等传感器时间	动态追踪、闭环控制

与之密切相关的还有 **机器人运动学标定** (kinematic calibration) 和 **相机-IMU 标定** (camera-IMU calibration)。

重要区分： RealSense 相机通常已有出厂内参；安装到 Tron2 上以后，仍需要确定相机相对于机器人运动学链的位置和朝向。对于可转动的头部云台，相机到基座的变换还是随关节角度变化的。

二、最重要的数学概念与坐标系

2.1 坐标系约定

符号	名称	说明
W	World frame	世界/仿真坐标系
B	Base frame	机器人基座坐标系
G	Gripper/end-effector frame	机械臂末端坐标系
C	Camera optical frame	相机光学坐标系
O	Object frame	物体坐标系
T	Target frame	标定板坐标系

以下统一采用明确的方向定义：

$${}^A T_B = \begin{bmatrix} {}^A R_B & {}^A t_B \\ 0 & 1 \end{bmatrix}$$

表示把在 B 中表达的点坐标变换为在 A 中表达的坐标。其中旋转 $R \in SO(3)$ ，平移 $t \in \mathbb{R}^3$ ，齐次刚体变换 $T \in SE(3)$ 。

三维点的齐次坐标满足

$$\bar{p}_A = {}^A T_B \bar{p}_B.$$

求逆与组合规则分别为

$${}^B T_A = ({}^A T_B)^{-1}, \quad {}^A T_C = {}^A T_B {}^B T_C.$$

例如 FoundationPose 得到物体在相机中的位姿 ${}^C T_O$ ，控制器想要物体在基座中的位姿，则有

$${}^B T_O = {}^B T_C {}^C T_O.$$

相机标定的关键任务之一，就是获取足够准确的 ${}^B T_C$ 。

2.2 机器人坐标系与相机 Optical Frame

ROS 常见约定 (REP 103) 如下：

轴	机器人机体坐标系	相机光学坐标系
X	前	图像右
Y	左	图像下
Z	上	朝相机前方

因此，`camera_link` 不一定等同于 `camera_optical_frame`。从普通相机机体坐标系到 optical frame，通常还有一段固定旋转。不要直接把 OpenCV 输出的光学坐标当作机器人机体坐标使用。参考：ROS REP 103。

三、相机内参标定 (Intrinsic Calibration)

3.1 相机矩阵和畸变

理想针孔模型中，内参矩阵是

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

其中 f_x, f_y 是以像素为单位的焦距， c_x, c_y 是主点 (principal point)。对已校正畸变的相机坐标 (X, Y, Z) ：

$$u = f_x \frac{X}{Z} + c_x, \quad v = f_y \frac{Y}{Z} + c_y.$$

真实相机还有镜头畸变。OpenCV 中常见的五参数模型为

$$D = [k_1, k_2, p_1, p_2, k_3],$$

前两类参数分别描述径向畸变和切向畸变；鱼眼等镜头需要相应的其他模型。

3.2 内参有什么用？

任务一：三维点投影到图像。给定相机坐标系中 $p_C = (X, Y, Z)$ ，利用 K 计算其像素位置 (u, v) 。用于可视化、重投影、视觉控制和仿真—真实图像对齐。

任务二：RGB-D 反投影。已知去畸变像素 (u, v) 及其光学轴深度 Z ，恢复

$$p_C = \begin{bmatrix} (u - c_x)Z/f_x \\ (v - c_y)Z/f_y \\ Z \end{bmatrix}.$$

如果使用原始畸变图像，应先去畸变或使用正确的反投影函数；如果深度图已经对齐到 RGB 图，则使用对齐后数据对应的相机模型。

任务三：6D 位姿估计。PnP、FoundationPose、MegaPose 和基于 CAD 模型的跟踪常使用内参。注意：仅凭普通 RGB 图像中的一个像素，无法独立恢复该点的绝对深度。

3.3 标定板选择

标定板	优点	注意事项	典型用途
Checkerboard (棋盘格)	精度高、成本低	需要足够清晰的角点	基础内参标定
ArUco / AprilTag	易检测、有 ID	单个标记角点精度受分辨率影响	PnP、位姿跟踪
ChArUco	棋盘角点 + ID，可应对部分遮挡	打印和角点对应关系需准确	高质量内参、手眼标定

对于普通机器人实验，优先建议 **ChArUco**；也可使用检测稳定的棋盘格。OpenCV 的 [ArUco/ChArUco 指南](#) 提供标板检测示例。

3.4 内参标定实际流程

- 制作标定板。**例如 7×5 方格的 ChArUco 板，方格边长 30 mm、标记边长 20 mm；这些只是示例。打印必须按真实物理尺寸，固定在刚性平板上，避免卷曲。
- 采集图像。**以 20–40 张清晰、有效图像作为初始数据集；让标定板出现在中心及四角，覆盖不同距离、倾斜角和尺度；固定对焦、曝光或其他关键成像设置。
- 检测角点。**可调用 `cv2.findChessboardCornersSB()` 或相应的 ChArUco 接口。确保 2D–3D 角点索引对应。
- 估计内参与畸变。**优化各帧相机姿态、 K 、 D ，最小化重投影残差：

$$\min_{K, D, \{R_j, t_j\}} \sum_{j,i} \|p_{ij} - \pi(K, D, R_j P_i + t_j)\|_2^2.$$

- 独立验证。**查看 RMS 重投影误差、图像各区域的残差、去畸变后的直线，以及未用于标定的图像。约 0.3–0.5 px 可以作为高质量标定的参考范围，但绝不是所有分辨率、镜头和任务通用的合格线，也不代表毫米级定位误差。

四、相机外参标定 (Extrinsic Calibration)

4.1 外参到底是什么？

相机内参回答“三维点怎样投影到图像”；外参回答“相机相对某个参考坐标系在哪里”。对于 Tron2 头部相机，一个典型目标是求

$${}^B T_C = \begin{bmatrix} {}^B R_C & {}^B t_C \\ 0 & 1 \end{bmatrix}.$$

4.2 已知标定板位姿时：PnP + 坐标变换

如果已经准确知道标定板在机器人基座中的位姿 ${}^B T_T$ ，通过相机内参和图像角点，PnP 可估计 ${}^C T_T$ 。于是

$${}^B T_T = {}^B T_C {}^C T_T,$$

从而

$${}^B T_C = {}^B T_T ({}^C T_T)^{-1}.$$

实操步骤：固定标定板；通过测量或机器人接触法确定 ${}^B T_T$ ；获取图像；检测对应角点并运行 PnP；计算 ${}^B T_C$ ；使用独立观测验证。

```
import cv2
import numpy as np

# N×3 的标定板三维角点，单位：m
object_points = np.asarray(..., dtype=np.float64)
# 对应的 N×2 图像角点，单位：pixel
image_points = np.asarray(..., dtype=np.float64)
K = np.asarray(..., dtype=np.float64)
dist = np.asarray(..., dtype=np.float64)

ok, rvec, tvec = cv2.solvePnP(
    object_points, image_points, K, dist
)
if not ok:
    raise RuntimeError("PnP failed")

R, _ = cv2.Rodrigues(rvec)
T_C_T = np.eye(4)
T_C_T[:3, :3] = R
T_C_T[:3, 3] = tvec.reshape(3)

# 必须来自独立测量，而不是凭空假设
T_B_T = np.asarray(..., dtype=np.float64)
T_B_C = T_B_T @ np.linalg.inv(T_C_T)
print(T_B_C)
```

方向提醒：solvePnP 返回的是标定板到相机的变换 ${}^C T_T$ ，不是其逆。

限制：如果标定板只是随意摆在桌面上，且不知道 ${}^B T_T$ ，一张图像通常不足以得到 ${}^B T_C$ ；此时可使用下一节的手眼标定。

五、手眼标定 (Hand-Eye Calibration)

5.1 两种常见布局

项目	Eye-in-Hand	Eye-to-Hand
相机	安装在机器人末端，随机机械臂运动	固定于基座/环境
标定板	通常固定在场景中	通常固定在末端
典型未知量	${}^G T_C$	${}^B T_C$
常见用途	腕部 RGB / RGB-D 相机	外部/顶视相机

注意，可随 yaw/pitch 转动的 Tron2 头部相机，并不是严格意义上的固定 Eye-to-Hand 相机；只能在云台姿态固定时暂时如此处理。

5.2 Eye-in-Hand: 腕部相机

希望求解固定变换 ${}^G T_C$ 。将标定板固定在桌上，驱动机械臂到不同姿态，每次记录：

- 由机器人 FK 或 TF 得到末端在基座中的位姿 ${}^B T_{G,i}$ ；
- 由 PnP 得到标定板在相机中的位姿 ${}^C T_{T,i}$ 。

因为标定板不动，有

$${}^B T_{G,i} {}^G T_C {}^C T_{T,i} = {}^B T_T.$$

不同采样构成经典的 $AX = XB$ 手眼标定问题，其中 $X = {}^G T_C$ 。需要包含多轴旋转的丰富运动；单纯沿三个方向平移，或只绕同一旋转轴转动，通常不足以充分约束问题。

采样建议：从约 20–30 个有效且角度多样的姿态开始，确保标定板可见、关节状态与图像同步，排除模糊和检测失败样本。姿态质量往往比机械增加样本数更重要。

OpenCV 包含 Tsai、Park、Horaud、Andreff、Daniilidis 等求解选项；参考 [calibrateHandEye](#)。

5.3 Eye-to-Hand: 固定相机、末端标定板

现在相机固定在环境中，标定板刚性安装在机器人末端。每次观察满足

$${}^B T_{G,i} {}^G T_T = {}^B T_C {}^C T_{T,i}.$$

其中相机位姿 ${}^B T_C$ 和末端—标定板位姿 ${}^G T_T$ 为固定未知量。根据多组姿态可求 ${}^B T_C$ ，而不必事先知道标定板放在桌面上的绝对位置。

关键：使用 `cv2.calibrateHandEye()` 做 Eye-to-Hand 时，机器人位姿输入应转换为 ${}^G T_B = ({}^B T_G)^{-1}$ ，标定板位姿仍输入 ${}^C T_T$ ；输出才可解释为 ${}^B T_C$ 。直接照搬 Eye-in-Hand 的输入可能造成错误的变换方向。

六、Tron2 头部与腕部相机的特殊情况

6.1 可转动头部相机

定义 H 为头部运动学链的末端安装坐标系， C_h 为头部相机光学坐标系， q_h 为头部关节角。则

$${}^B T_{C_h}(q_h) = {}^B T_H(q_h) {}^H T_{C_h}.$$

- ${}^B T_H(q_h)$ ：头部运动学模型和关节角决定的**动态变换**；
- ${}^H T_{C_h}$ ：相机安装支架决定的**固定变换**，需要标定或测量。

推荐方法：把 ChArUco 板固定在桌面；采集多个 yaw/pitch 姿态的图像和关节角；用 FK 求 ${}^B T_H$ ，由 PnP 求 ${}^C_h T_T$ ，利用 Eye-in-Hand 的形式求 ${}^H T_{C_h}$ ；最后用保留的姿态验证。

yaw/pitch 旋转轴不相交时

不能简单地写成 $R_z(q_y)R_y(q_p)$ 并忽略关节中心之间的平移。应使用完整的刚体变换链：

$${}^B T_{C_h} = {}^B T_Y T_Y(q_y) {}^Y T_P T_P(q_p) {}^P T_{C_h}.$$

其中 ${}^Y T_P$ 包含 yaw、pitch 轴之间的固定旋转和平移。优先核查 URDF 里的每个关节 `origin`、`axis`。如果机械臂/云台运动学本身有系统性偏差，单独修正相机安装外参不一定能完全消除各个姿态下的误差。

对于当前固定头部视角的螺丝刀抓取实验，可先固定云台姿态完成一组基座—相机外参，再逐步扩展到动态头部模型。

6.2 腕部相机

定义 C_w 为腕部相机, q_a 为机械臂关节角, 则

$${}^B T_{C_w}(q_a) = {}^B T_G(q_a) {}^G T_{C_w}.$$

其中 ${}^B T_G(q_a)$ 来自 FK, ${}^G T_{C_w}$ 由 Eye-in-Hand 标定。若腕相机安装在 `wrist_link` 而非用于运动学记录的 G 坐标系, 应使用一致的链接并正确组合固定变换。

6.3 头部相机与腕部相机之间的变换

一旦两台相机都在机器人基座中有位姿, 就可以计算:

$${}^{C_h} T_{C_w} = ({}^B T_{C_h})^{-1} {}^B T_{C_w}.$$

因此, 两台相机不一定要单独做一轮“双目标定”; 但通过共同观察标定板来验证两路外参与运动学的一致性仍然非常有价值。由于两条运动学链会运动, ${}^{C_h} T_{C_w}$ 通常**不是常数**。

七、RGB-D 深度相机标定与对齐

7.1 设备内部坐标系

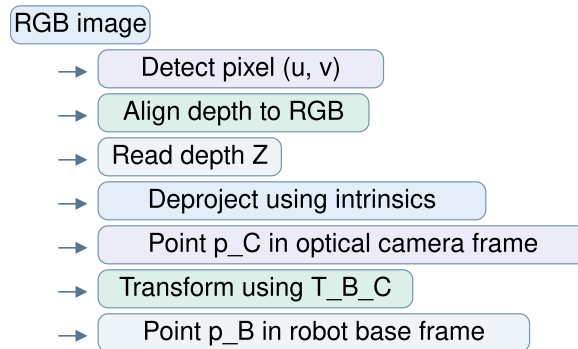
D435、D455 等 RGB-D 设备通常包含 RGB、左右红外/深度等不同的光学坐标系。RGB 与 Depth 的光学中心通常不完全重合, 因而需要已标定的

$${}^{C_{\text{rgb}}} T_{C_{\text{depth}}}.$$

RealSense SDK 一般能够读取相机出厂标定的内参、外参和深度尺度, 通常先**读取并验证**即可。

7.2 RGB 像素到机器人三维点

基本流水线:



在 RealSense SDK 中, 对齐与反投影的一个示例如下:

```

import pyrealsense2 as rs
import numpy as np

pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.depth)
config.enable_stream(rs.stream.color)
pipeline.start(config)

```

```

align = rs.align(rs.stream.color)
try:
    frames = pipeline.wait_for_frames()
    aligned = align.process(frames)

    depth = aligned.get_depth_frame()
    color = aligned.get_color_frame()
    if not depth or not color:
        raise RuntimeError("Missing RGB-D frame")

    u, v = 320, 240 # 示例像素, 须检查图像边界
    z = depth.get_distance(u, v) # 米
    if z <= 0:
        raise RuntimeError("Invalid depth")

    intr = depth.profile.as_video_stream_profile().get_intrinsics()
    point_C = rs.rs2_deproject_pixel_to_point(
        intr, [float(u), float(v)], z
    )
    print("Point in camera optical frame:", point_C)
finally:
    pipeline.stop()

```

随后使用已经标定的相机到基座变换:

```

point_C_h = np.r_[np.asarray(point_C), 1.0]
point_B_h = T_B_C @ point_C_h
print("Point in base frame:", point_B_h[:3])

```

这就完成了“RGB 像素 → 相机三维点 → 机器人基座三维点”。

7.3 深度相机需要重新标定吗?

先确认出厂数据和实际表现, 再决定是否进行设备的深度重新标定。建议检查:

检查项	目的
RGB 内参	保证 RGB 像素投影正确
Depth 内参	保证深度反投影正确
RGB-Depth 外参	保证对齐准确
深度尺度	确保单位正确
目标工作距离处的深度精度	确保抓取距离可靠
相机-机器人外参	确保视觉点能正确进入基座坐标系

对于螺丝刀等细长或反光物体, 要注意深度空洞、边缘错配和多路径/反光等误差; 它们不一定能通过重新标定内参解决。参考: RealSense SDK。

八、OpenCV 完整手眼标定代码

8.1 数据格式

每个姿态记录两组匹配的 4×4 齐次变换, 并以同一时间点对应:

```

base_T_gripper.npy  shape = (N, 4, 4)
camera_T_target.npy shape = (N, 4, 4)

```

即分别存放 ${}^B T_{G,i}$ 和 ${}^C T_{T,i}$, 平移采用米。保存的矩阵应满足旋转正交、行列式约等于 1, 底行等于 $[0, 0, 0, 1]$ 。

8.2 统一支持 Eye-in-Hand / Eye-to-Hand

```
import cv2
import numpy as np

# ^B T_G : 机器人末端 -> 机器人基座
T_B_G = np.load("base_T_gripper.npy")
# ^C T_T : 标定板 -> 相机光学坐标系
T_C_T = np.load("camera_T_target.npy")

assert T_B_G.shape == T_C_T.shape
assert T_B_G.ndim == 3 and T_B_G.shape[1:] == (4, 4)
assert len(T_B_G) >= 3

mode = "eye_in_hand" # 或 "eye_to_hand"

if mode == "eye_in_hand":
    robot_poses = T_B_G
elif mode == "eye_to_hand":
    robot_poses = np.linalg.inv(T_B_G)
else:
    raise ValueError("Unknown calibration mode")

R_robot = [T[:3, :3] for T in robot_poses]
t_robot = [T[:3, 3] for T in robot_poses]
R_target = [T[:3, :3] for T in T_C_T]
t_target = [T[:3, 3] for T in T_C_T]

R, t = cv2.calibrateHandEye(
    R_robot, t_robot, R_target, t_target,
    method=cv2.CALIB_HAND_EYE_PARK
)

X = np.eye(4)
X[:3, :3] = R
X[:3, 3] = t.reshape(3)

if mode == "eye_in_hand":
    print("^G T_C (gripper <- camera):")
else:
    print("^B T_C (base <- camera):")
print(X)
np.save("handeye_result.npy", X)
```

该示例展示接口用法，但实际部署前还需要剔除坏样本、检查运动激励是否充分，并在独立姿态上验证结果。尤其必须确保 camera_T_target 使用的是光学相机坐标系而非 camera_link。

8.3 标定结果验证

Eye-in-Hand: 对每一组采样计算

$$\hat{T}_{BT,i} = {}^B T_{G,i} {}^G T_C {}^C T_{T,i}.$$

如果目标板真的固定，所有 $\hat{T}_{BT,i}$ 应彼此接近。

Eye-to-Hand: 对每一组采样计算

$$\hat{T}_{GT,i} = ({}^B T_{G,i})^{-1} {}^B T_C {}^C T_{T,i}.$$

如果标定板刚性安装在末端，所有 $\hat{T}_{GT,i}$ 应彼此接近。

实践中应分别报告平移残差、旋转角残差，并保留未参加拟合的姿态做独立测试，而不能只目测最终矩阵是否“差不多”。

九、ROS 2 中的标定工具与基本操作

9.1 常用工具

工具	功能
camera_calibration	基于棋盘格的相机内参标定
image_pipeline	图像去畸变和校正
tf2	机器人坐标变换管理
robot_state_publisher	用 URDF 和关节角发布 TF
easy_handeye2	ROS 2 手眼标定 workflow
MoveIt Calibration	手眼标定 GUI
Kalibr	多相机/相机-IMU 时空标定

9.2 相机内参标定

在支持的 ROS 2 发行版中：

```
sudo apt install \
  ros-${ROS_DISTRO}-camera-calibration \
  ros-${ROS_DISTRO}-image-pipeline

ros2 topic list
```

假设相机 Topic 为 /camera/image_raw、/camera/camera_info，可运行：

```
ros2 run camera_calibration cameracalibrator \
  --size 8x6 \
  --square 0.025 \
  image:=/camera/image_raw \
  camera:=/camera
```

这里 8x6 是内部角点数，0.025 表示方格边长为 0.025 m；实际参数必须与标定板一致。不同相机驱动的 Topic 名称可能不同。参考：image_pipeline。

9.3 ROS 2 手眼标定

可参考 easy_handeye2 或 MoveIt Calibration。典型 workflow 是：

1. ROS 发布机器人末端 TF；
2. 检测目标板并发布标定板 TF；
3. 在多个机器人姿态下采样；
4. 选择 Eye-in-Hand / Eye-to-Hand；
5. 计算、验证并保存手眼结果；
6. 将结果接入机器人 URDF/TF 图。

只需要一次性计算时，OpenCV Python 脚本通常已足够；反复做团队实验时，ROS 工具更有利于标准化。

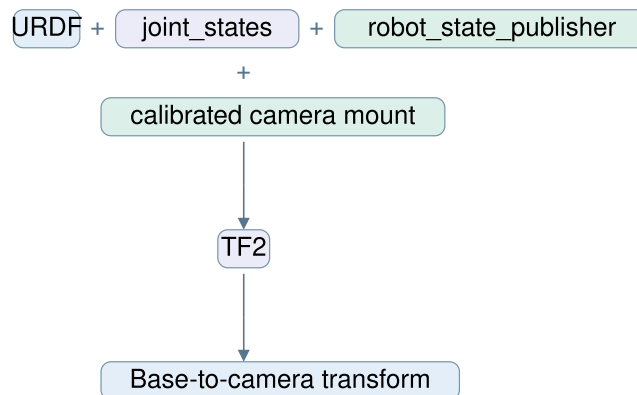
十、将标定结果写入 ROS TF2 / URDF

10.1 推荐的 TF 结构

以下仅为示例，具体链接名字以 Tron2 URDF 为准：

```
base_link
├── arm_link_1
│   └── ...
│       ├── wrist_link
│       │   ├── wrist_camera_link
│       │   └── wrist_camera_optical_frame
└── head_yaw_link
    ├── head_pitch_link
    │   ├── head_camera_link
    │   └── head_camera_optical_frame
```

相机固定安装外参与腕部/头部关节的动态 FK 分开：



10.2 静态外参发布

下面的平移与旋转只是**命令格式示意**，不是 Tron2 标定数据：

```
ros2 run tf2_ros static_transform_publisher \
  --x 0.03 --y 0.00 --z 0.08 \
  --qx 0 --qy 0 --qz 0 --qw 1 \
  --frame-id wrist_link \
  --child-frame-id wrist_camera_optical_frame
```

注意：实际光学坐标系的旋转一般不是单位四元数。若变换已由 URDF + robot_state_publisher 发布，不要重复发布相同的父子 TF 边。

10.3 动态 TF 检查

```
ros2 run tf2_ros tf2_echo \
  base_link wrist_camera_optical_frame

ros2 run tf2_ros tf2_echo \
  base_link head_camera_optical_frame
```

机器人关节运动时，相机相对于基座的变换应该随之更新。对采集时的图像，查询 TF 应使用**图像时间戳**，而不是不加区分地取机器人最新状态。

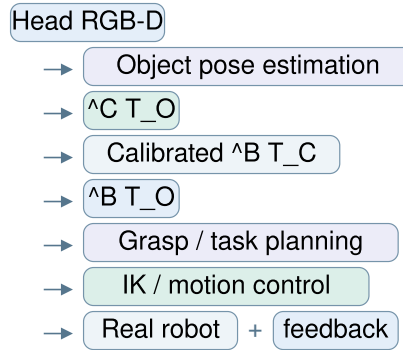
十一、相机标定如何进入机器人学习

11.1 物体 6D 位姿估计与视觉抓取

假设 FoundationPose 从头部 RGB-D 估计到螺丝刀位姿 ${}^C{}_hT_O$ ，那么

$${}^B T_O = {}^B T_{C_h} (q_h) {}^C{}_hT_O.$$

流程：



必须区分视觉估计器误差与相机外参误差。即使 6D Pose Estimator 很准，错误的 ${}^B T_C$ 也会带来目标在基座下的系统性偏移。

11.2 模仿学习与 VLA

动作可能以 TCP 增量 $a_t = \Delta T_{TCP}$ ，或者关节序列 $a_t = q_{t:t+H}$ 表示。是否显式依赖外参取决于任务：

任务	是否需要显式外参
RGB 到关节命令的端到端行为克隆	不一定
RGB + proprioception 的端到端策略	不一定
通过像素与深度恢复三维物体位置	需要
视频轨迹重定向为机器人 SE(3) 轨迹	通常需要
多相机的三维融合	需要
仿真/真机相机几何对齐	需要
FoundationPose + IK	需要

端到端 VLA 可以隐式学习视觉-动作映射，但准确、统一的几何关系通常有助于跨相机、跨机器人和跨场景迁移。本节只回答**何时需要显式外参**；BC/ACT 和示教数据处理见[模仿学习](#)，SAM/Foundation 系列、时序滤波与失踪恢复见[视觉状态估计](#)。

11.3 强化学习与 Sim-to-Real

视觉策略可以写成

$$a_t = \pi(I_t, q_t).$$

若仿真与真机的相机位置、视角、内参、裁剪/缩放方式不同，便会引入观测分布偏移。应优先对齐：

$$K_{\text{sim}} \approx K_{\text{real}}, \quad {}^B T_{C,\text{sim}} \approx {}^B T_{C,\text{real}}.$$

之后可在训练时对残余标定误差施加 Camera Domain Randomization，例如

$$\delta t \sim \mathcal{N}(0, \Sigma_t), \quad \delta \theta \sim \mathcal{N}(0, \Sigma_R).$$

随机化用于增强鲁棒性，不应替代尽可能准确的基础标定。

11.4 视觉闭环控制与时间同步

对目标物体和当前物体位姿，可以构造 $SE(3)$ 位姿误差：

$$e_t = \text{Log}((T_O^*)^{-1}T_{O,t}).$$

但实时视觉闭环的关键不只有几何外参，还包括**时间对齐**。若图像曝光发生在 t_{image} ，应使用同一时刻的关节状态计算

$${}^B T_C(t_{\text{image}}),$$

而不是随意使用当前最新机器人姿态。移动中的头部/腕部相机尤为如此。

十二、标定精度与结果验证

必须同时评估图像层、几何层与任务层的质量。

验证层次	关键指标	目的
图像层	重投影 RMS / 分布	检查内参与目标板角点
几何层	平移残差、旋转角残差	检查外参与手眼约束
任务层	三维定位误差、抓取成功率	检查整套机器人闭环系统

12.1 重投影误差

检测角点 p_i 与投影预测 $\hat{p}_i$ 之间的欧氏误差为

$$e_i = \|p_i - \hat{p}_i\|_2,$$

其 RMS 为

$$e_{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{i=1}^N e_i^2}.$$

图像里的像素误差不能直接等同于机器人末端的毫米级误差。

12.2 三维定位误差

对真实位置已知的测试点 p_B^* ，比较视觉估计 $\hat{p}_B$ ：

$$e_p = \|\hat{p}_B - p_B^*\|_2.$$

分别测试工作空间中心、边缘、不同高度、不同观察方向，以及不同 yaw/pitch 姿态。如果头部一转动误差就显著增大，应优先检查头部运动学、转轴位置、时间同步和相机安装变换。

12.3 实际抓取测试

还可以检查 TCP 的目标位置误差

$$e_{\text{grasp}} = \|p_{\text{TCP,actual}} - p_{\text{TCP,target}}\|_2$$

及螺丝刀抓取成功率、重复性、目标姿态误差和扰动恢复能力。要注意这些指标还受到 FK、IK、执行器、TCP 几何和物体位姿估计的共同影响；抓取失败**不等于**相机标定失败。

十三、常见问题与排查

现象	可能原因	排查方式
目标位置整体偏移	外参平移偏差	检查 ${}^B T_C$
误差随视角增加	旋转外参或运动学误差	多关节姿态验证
头部旋转后漂移	yaw/pitch 轴与偏移建模不准确	检查 URDF/实际机构
腕部手眼标定不稳定	旋转激励不足或坏样本	增加多轴姿态、剔除异常值
深度距离不准	深度尺度、反光或深度噪声	测量已知距离
RGB 与深度图错位	对齐或帧模型错误	检查 alignment 和内参
图像方向与机器人方向不一致	optical frame 混用	检查 TF 坐标轴
机器人运动时标定误差变大	图像与关节时间戳不一致	查询曝光时刻的 TF
仿真正常、真机失败	相机/动力学/控制存在 gap	分别验证视觉与执行

手眼采样的简单保守策略是：**运动到位** → **等待稳定** → **同时记录图像和对应机器人状态** → **保存样本**。这样可以减少运动模糊、时间偏移及末端弹性形变影响。

十四、推荐开源工具与官方资料

工具	主要功能	使用建议
OpenCV	内参、PnP、手眼标定	优先掌握
ROS image_pipeline	ROS 内参及图像处理	使用 ROS 时推荐
easy_handeye2	ROS 2 手眼标定	可用于可重复 workflow
MoveIt Calibration	GUI 标定流程	需要可视化交互时
Kalibr	多相机、相机-IMU、时间标定	复杂多传感器系统
RealSense SDK	设备参数、深度对齐、反投影	D435/D455 等设备
ROS REP 103	ROS 坐标系规范	检查 optical frame

对于只有头部和腕部两台相机的操作系统，不必一开始就引入复杂的相机-IMU 联合标定工具。优先把机器人 TF 链、PnP 和手眼标定做好。

十五、针对 Tron2 的实施路线

建议分成四个阶段进行，避免一开始把所有误差来源混合在一起。

阶段一：设备内参与深度检查

- 读取 RealSense RGB/Depth 内参和畸变；
- 对腕部 RGB 图像进行必要的内参验证或重标定；
- 检查 RGB-Depth 对齐、深度尺度和典型抓取距离上的误差；
- 确认图像分辨率、裁剪及内参与视觉模型输入匹配。

此阶段通常不需要机械臂运动。

阶段二：头部相机几何标定

目标：获得 ${}^B T_{C_h}(q_h)$ 。

1. 核查 Tron2 的 yaw/pitch 关节轴、固定偏移和 URDF；
2. 固定 ChArUco 板，采集多个云台姿态；
3. 记录对应 $q_{h,i}$ ，使用 FK 求 ${}^B T_H(q_{h,i})$ ；
4. 使用 PnP 求 ${}^{C_h} T_{T,i}$ ；
5. 标定固定相机安装变换 ${}^H T_{C_h}$ ；
6. 在未参加求解的 yaw/pitch 姿态下测试三维定位误差。

若当前只使用一个头部角度，可先将云台固定并完成简化版标定。

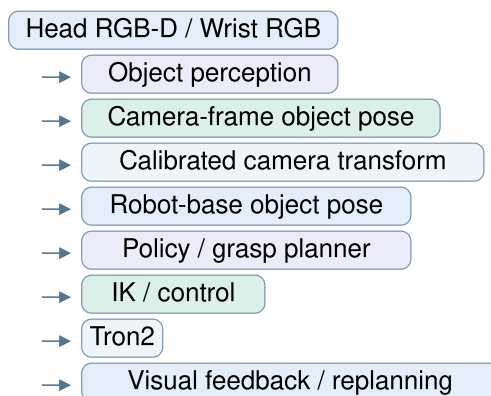
阶段三：腕部相机与 TCP

目标：获得 ${}^G T_{C_w}$ 和 ${}^G T_{TCP}$ 。

1. 固定标定板，在 20–30 个多样姿态下采样；
2. 保存末端 FK 与相机看到的板位姿；
3. 运行 `calibrateHandEye()`；
4. 用未见姿态验证固定目标板的一致性；

阶段四：视觉闭环与学习部署

统一数据流：



优先完成三个关键变换：

$$\boxed{{}^B T_{C_h}(q_h)}, \quad \boxed{{}^G T_{C_w}}, \quad \boxed{{}^G T_{TCP}}.$$

得到这三个结果后，再结合 FK、TF2 和时间同步，即可把头部感知、腕部视觉及抓取控制整合到统一坐标系统中。

十六、总结与检查清单

核心问题	对应技术	机器人学习中的作用
图像如何对应三维几何？	内参与畸变、PnP、RGB-D	深度反投影、位姿估计
相机相对于机器人在哪里？	外参、手眼标定	抓取坐标变换
机器人运动后变换是否仍然有效？	FK、URDF、TF2、时间同步	主动视觉、视觉闭环

实际部署前建议逐项确认：

- ☐ 使用正确的图像分辨率与内参，并明确像素是否已去畸变。
- ☐ 所有变换都标清方向（如 ${}^B T_C$ 而非模糊的 `camera_extrinsic`）。
- ☐ 确认 `camera_link` 与 `camera_optical_frame` 的区别。
- ☐ 所有机器人 FK、PnP、手眼采样使用一致的长度单位和帧定义。
- ☐ 头部 yaw/pitch 轴、关节固定偏移以及腕部相机安装外参正确。
- ☐ RGB-D 已对齐，且知道深度代表的轴向距离和尺度。
- ☐ 对外参使用独立的多姿态数据验证，而非只看训练数据残差。
- ☐ 动态实验时，图像与关节角/TF 时间戳严格对应。
- ☐ 最终使用三维位置测试和真实抓取表现闭环验证。

对于 Tron2 操作系统，最值得优先掌握的是 OpenCV PnP、Hand-Eye Calibration、机器人 FK 和 ROS TF2。这四项结合起来，就能构建不依赖 Vicon、适合头部/腕部视觉感知的相机标定基础设施。

说明：本教程中的尺寸、采样数量、路径、TF 名称及部分命令参数为示例或工程建议，并非 Tron2 的厂家参数或经现场标定的真实结果。实际部署必须以设备、URDF、采集日志与实验验证为准。