Python version :3.10.6??学校的是几阿

Starting with Python 3.1, Python (on most systems) is now able to choose the shortest of these and simply display 0.1.

有要求一定要参考多少本书吗

with open https://blog.csdn.net/weixin\_39613951/article/details/110785735

连续多个open

误差： 考虑数据向量是frame到frame的，要加上visual的高度吗，再与原来的作对比，看看原来的gripper会顶到rot plate吗

图像坐标转显示坐标： ！！！

https://juejin.cn/s/opencv%20%E5%9B%BE%E5%83%8F%E5%9D%90%E6%A0%87%E8%BD%AC%E6%8D%A2%E7%8E%B0%E5%AE%9E%E5%9D%90%E6%A0%87

照相机标定。。。

看看别的编程类的论文

问老师：

之前的是urdf现在是xacro， 用xacro来写没问题把

wir, unser 的使用

link frame ist entschied in cad?

写的时候直接看 + 找外文链接

自动对焦：

https://github.com/huchunxu/ros\_exploring/blob/master/robot\_marm/marm\_planning/scripts/trajectory\_demo.py

https://answers.ros.org/question/50610/the-meaning-of-velocities-accelerations-and-time\_from\_start-in-jointtrajectorypointmsg/

https://www.cnblogs.com/feifanrensheng/p/15579537.html

https://www.programcreek.com/python/example/123227/trajectory\_msgs.msg.JointTrajectoryPoint

trajectory：

http://wiki.ros.org/Robots/TIAGo/Tutorials/trajectory\_controller

报错：eg：path error：

http://docs.ros.org/en/noetic/api/control\_msgs/html/action/FollowJointTrajectory.html

clock不需要调，直接用服务端的：

http://wiki.ros.org/Clock

clock：

https://design.ros2.org/articles/clock\_and\_time.html

周五：

two ways...

http://wiki.ros.org/joint\_trajectory\_controller#ROS\_API

http://wiki.ros.org/ros2\_controllers/Tutorials

vel,acc = 0 , stabil:

https://answers.ros.org/question/50610/the-meaning-of-velocities-accelerations-and-time\_from\_start-in-jointtrajectorypointmsg/

背景：（其他的也可以写背景）

https://zhuanlan.zhihu.com/p/467256942

jointpoint单位：

https://docs.ros2.org/foxy/api/trajectory\_msgs/msg/JointTrajectoryPoint.html

python opencv找椭圆

https://zhuanlan.zhihu.com/p/603629524

https://blog.csdn.net/Useless\_csdn/article/details/102616576

https://zhuanlan.zhihu.com/p/438542999

https://zhuanlan.zhihu.com/p/603629524

https://blog.csdn.net/Useless\_csdn/article/details/102616576

椭圆拟合3种方法：

https://blog.csdn.net/weixin\_41616991/article/details/126158231

看获得的图片是几通道的：

https://stackoverflow.com/questions/19062875/how-to-get-the-number-of-channels-from-an-image-in-opencv-2

https://numpy.org/doc/stable/reference/generated/numpy.ndarray.ndim.html#numpy-ndarray-ndim

注意调参：

https://bbs.huaweicloud.com/blogs/251166

https://blog.51cto.com/u\_15362367/3888720

https://blog.csdn.net/HaoZiHuang/article/details/106437761

circles = cv2.HoughCircles(

edges, cv2.HOUGH\_GRADIENT, 1, 50, param1=100, param2=30, minRadius=5, maxRadius=100)

会 return none

https://stackoverflow.com/questions/31140386/python-opencv-hough-circles-returns-none

https://www.google.com/search?q=cv2.HoughCircles+return+None&client=ubuntu-sn&hs=MAz&channel=fs&ei=xfxUZM7ANdCIxc8Puoi58As&ved=0ahUKEwiOt4nwnN7-AhVQRPEDHTpEDr4Q4dUDCA8&oq=cv2.HoughCircles+return+None&gs\_lcp=Cgxnd3Mtd2l6LXNlcnAQDDIGCAAQFhAeOhMIABCPARDqAhC0AhCMAxDlAhgBOhMILhCPARDqAhC0AhCMAxDlAhgBSgQIQRgAUJoKWJoKYK0RaAFwAXgAgAE-iAE-kgEBMZgBAKABAaABArABCsABAdoBBAgBGAo&sclient=gws-wiz-serp

https://stackoverflow.com/questions/50568668/understanding-houghcircles-in-python-opencv-cv2

if circles is not None:。。。

cv2.画圆要是int：https://docs.opencv.org/2.4/modules/core/doc/drawing\_functions.html#circle

图片坐标：https://blog.csdn.net/oqqENvY12/article/details/71933651

https://blog.csdn.net/lz0499/article/details/80978433

findcontur 可能因为原图不圆导致质心不是圆心，但是可以调教优化原图，但尽管如此如果原图的圆存在缺陷就不行了。（例如有灰尘遮挡） 有对比截图

cv2.moments: https://xie.infoq.cn/article/c61a7c1d9547813a319042879

fincontur返回值：在看别的

https://juejin.cn/s/cv2.findcontours%20%E8%BF%94%E5%9B%9E%E5%80%BC

使用Python的OpenCV获取二值图像轮廓及中心点坐标的代码：chat机器人

经过滤波之后的casd 坐标与asd不太一样 。。那用或不用别的呢，有啥影响

\*\*\* 主要是如何评判哪个结果好啊 不知道 \*\*\*

urdf的修改 摄像头？ teil\_change里1k 2k是直接改数据的， 想一想阿 想不起来了

in opencv gibt es 圆检测的函数， but我们不用， 用轮廓检测

放不放大的图片 点坐标不一样

moments 解释:

https://docs.opencv.org/4.x/dd/d49/tutorial\_py\_contour\_features.html

https://en.wikipedia.org/wiki/Image\_moment

相同的图片 houghcirlce和 findcontur作对比： 精度应该差不多，但是 findcontur更稳定

findcontur会修改图像输入.. https://www.cnblogs.com/zyly/p/9327425.html

3.2以后就不会了..

！！！ https://blog.csdn.net/u012566751/article/details/54379290

以及下面的下载文件

高精度定位

这个参数还行：

https://blog.csdn.net/kingroc/article/details/86167222

根据要求知道要修改的是什么标签了，但是在编写程序之前，我们需要先探究一下改动这些属性，对其他urdf标签有什么影响。以及相邻的link joint

http://wiki.ros.org/Papers/TePRA2013\_Foote?action=AttachFile&do=get&target=TePRA2013\_Foote.pdf

前景motivation: studienarbeit纸上allgeimeins.

Die Aufgabe ist…… Das erste zu kalibrierte Batuteil ist 1K…

‘frame’… + Bild in ros/xml

Ein URDF(xacro) Dateil in ros ist xml sepzif… http://wiki.ros.org/urdf/XML

(Es besetht aus…) Die in (unsere) xacro Dateil verwendete(n) Elemente sind: (‘’)robot, link, joint, gazebo. Die Elemente link, joint, gazebo sind gekapselt in dem Element robot. Das Element link beschreibt die Eigenschaften der Bauteile eines Roboters. Das Element gazebo (ist URDF robot description format), und beschreibt die Performance im Simulator Gazebo, deswegen werden diese zwei Elemente nicht in Betracht gezogen.

(Lassen wir diskutieren..)Das Element joint ‘(describes the kinematics and dynamics of the joint and also specifies the safety limits of the joint. )’. Ein ‘joint’ Element hat zwei Attribute: name und type. ‘name’ gibt einen eindeutigen Name von joint an und ‘type’ gibt den Bewegungstyp an. Es besteht aus den folgenden untergeordneten Elenmenten: origin, parent, child, axis, calibration , dynamics, limit, mimic und safety\_controller. Die zu unserem Roboter relevant sind: origin, parent, child, axis und limit. In ‘parent’ Element wird der name des übergeordneten Links(parent link) geschrieben. In ‘child’ Element wird der name des untergeordneten Links(child link) geschrieben. Das ‘origin’ Element beschreibt die Transformation vom parent (Link) zum child Link. Es enthält zwei Attribute: xyz und rpy. ‘xyz’ repräsentiert den x-, y-, z-Offset und rpy repräsentiert die Rotation um die x-, y-, z-Achse. 'axis' gibt das Bezugssystem an. ‘limit’ legt die Obergrenze, die Untergrenze, die maximale Arbeit und die maximale Geschwindigkeit eines ‘joint’ fest.

In dieser Aufgabe wird ‘frame’ von den Bauteilen angepasst, dementsprechend ist der zu veränderte Teil das ‘origin’ Element. Werden die Elemente in einer URDF Dateil betrachtet, kann man daraus finden, dass die Elemente in ‘joint’ beeinflusst die Element in ‘link’ nicht(….) Deswegen sind nur die entsprechende Teile zu modifizieren, und die andere relevante Elemente nicht betrachten zu können.

Es ist ungünstig wenn wir TF verwenden. (urdf to graphiz darstellen)

(Zuerst) wird der 1K\_Dispenser betrachtet. Er ist ein ‘link’, der gleiten aber sich nicht drehen kann.

Dann wird … betrachtet

Ein urdf Dateil enthält die folgende

Decimal 模块可以直接实现7位以内的小数相加减同时不会增加小数

https://docs.python.org/3.11/tutorial/floatingpoint.html

[https://docs.python.org/3.11/library/string.html#formatstrings](https://docs.python.org/3.11/library/string.html" \l "formatstrings)

[https://docs.python.org/3.11/tutorial/inputoutput.html#old-string-formatting](https://docs.python.org/3.11/tutorial/inputoutput.html" \l "old-string-formatting)

https://anh.cs.luc.edu/handsonPythonTutorial/float.html

[https://docs.python.org/zh-cn/3/library/decimal.html#quick-start-tutorial](https://docs.python.org/zh-cn/3/library/decimal.html" \l "quick-start-tutorial)

https://blog.csdn.net/adorable\_/article/details/109577331

是float64吗 double类型？

Getcontxt.pre 只会影响decimal对象

二进制不能准确的表示数:

print(0.0795000000000249+0.0795000000000246)

print('%.17f'%(10.0795000000000249+10.0795000000000246))

0.1590000000000495

20.15900000000004866

17位有效数字截断，https://help.uis.cam.ac.uk/system/files/documents/python3-notes.pdf

我们的最大的不超过17..

print(0.07750000000002338+0.07750000000002337)

0.15500000000004677

print('%.18f'%(0.07750000000002338+0.07750000000002337))

0.155000000000046767

print(0.07950000000000249+0.07950000000000246)

print('%.19f'%(0.0795000000000249+0.0795000000000242))

print('%.19f'%(10.0795000000000249+10.0795000000000242))

0.15900000000000494

0.1590000000000491021

20.1590000000000486580

print(0.07950000000002435)

0.07950000000002436

print(0.07950000000002435+0.23156418)

0.31106418000002434

由于这些 长的加长的不准，长的加短的也不准，所以我们干脆直接全用decimal

(decimal buneng + float)

localcontext

print(Decimal('0.07950000000002435')+Decimal('0.23156418'))

print(Decimal(0.23156418))

print('%.53f'%(0.23156418))

0.31106418000002435 是python3.11之后才能Deciamal(float)的吗

0.23156418000000000834148750072927214205265045166015625 实际显示的，我们看到的数不是他真实的数 但是我们要的就是看到的数。。

0.23156418000000000834148750072927214205265045166015625 python会三选1 但这不是以前版本的python才会这样吗

程序里print(getcontext()) pre是28,所以rounding参数不需要round\_down。看看别的参数

程序里的Decimal()里得要是str，float的话会出事：修改完的数据会变成28位的，比如1K\_Dispenser。

程序里格式化用%s不是%f或者%.17f，是因为会出事。 str（0.31641532...)里的是float，本身就是不对的，所以str完还是不对，但是这里%s可以，哦，str是将他转化成字符串, 不对，直接%s（。。。）了。

反思： 所有的圆心组成的圆确定后，找到中心(rot\_plate) 移动 rot\_plate 到中心，如果转动之后loch中心仍在一个圆上，则准确。

展望： 编程不太行，编程思想可能不太对

ros为什么用四元数