3. Astra object tracking

3.1. Introduction

Package: /home/yahboom/YBAMR-COBOT-EDU-00001/src/yahboom_navrobo_astra

Official website: https://learnopencv.com/object-tracking-using-opencv-cpp-python/#opencv-tracking-api

- Object Tracking

Object tracking is the process of locating an object in consecutive video frames. This definition sounds straightforward, but in computer vision and machine learning, tracking is a very broad term that encompasses conceptually similar but technically different concepts. For example, all of the following different but related ideas are usually studied under object tracking:

(1) Dense Optical Flow (DOF): These algorithms help estimate the motion vector of each pixel in a video frame.

(2) Sparse optical flow: For example, the Kanade-Lucas-Tomashi (KLT) feature tracking algorithm tracks the positions of several feature points in an image.

(3) Kalman Filtering: A very popular signal processing algorithm based on prior motion information, used to predict the position of a moving target. One of the early applications of this algorithm was missile guidance! The onboard computer that guided the Apollo 11 lunar module to the moon had a Kalman filter.

(4) Meanshift and Camshift: These are algorithms for locating the maximum value of a density function, and they are also used for tracking.

(5) Single object trackers: In this type of tracker, the first frame is marked with a rectangular marker to indicate the position of the object to be tracked. A tracking algorithm is then used to track the object in subsequent frames. In most practical applications, these trackers are used in conjunction with object detectors.

(6) Multiple object track finding algorithms: When we have a fast object detector, it makes sense to detect multiple objects in each frame and then run a track finding algorithm to identify which rectangle in one frame corresponds to the rectangle in the next frame.

- Comparison of algorithms

Algorithm	Speed	Accuracy	Description
BOOSTING	Slow	Poor	Same machine learning algorithm as Haar casades (AdaBoost), but it has been around for more than a decade and is a veteran algorithm.
MIL	Slow	Poor	More accurate than BOOSTING, but with a higher failure rate.
KCF	Fast	High	Faster than both BOOSTING and MIL, but performs poorly in the presence of occlusion.
TLD	Average	Average	Very many false positives and serious crosstalk.
MEDIANFLOW	General+	General	Few false positives. The model will fail for fast-jumping or fast-moving objects.
GOTURN	General	General	A deep learning-based target detector that requires an additional model to run.
MOSSE	Fastest	High-	Really fast, but not as accurate as CSRT and KCF. If you want speed, you can choose it.
CSRT	Fast-	Highest	Slightly more accurate than KCF, but not as fast as KCF.

3.2, KCF object tracking

The full name of KCF is Kernel Correlation Filter. It was proposed by Joao F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista in 2014. The algorithm caused a sensation after it came out. This algorithm has very impressive performance in both tracking effect and tracking speed, so it has attracted a large number of scholars to study this algorithm and the industry has also gradually applied this algorithm in actual scenarios. This algorithm homepage contains papers and codes that can be downloaded here, as well as some introductions. This article was published on TPAMI by the author in 2015, so you may see two versions, but there are no changes, both can be seen. Paper download address The correlation filter algorithm is a discriminative tracking, which mainly uses the given samples to train a discriminative classifier to determine whether the tracked target is the surrounding background information. The sample is collected mainly by the rotation matrix, and the algorithm is accelerated by the fast Fourier transform.

3.2.1. Usage

Note: The [SWB] mid-range of the aircraft model remote control has the [emergency stop] function of this gameplay. Please put the aircraft model remote control in a convenient place for control. Pay attention to safety when playing! ! !

- To start control, you need to first turn the SWB button to the upper gear position (control command mode) to release the remote control

Turn off the self-starting chassis service

```
sudo supervisorctl stop ChassisServer
```

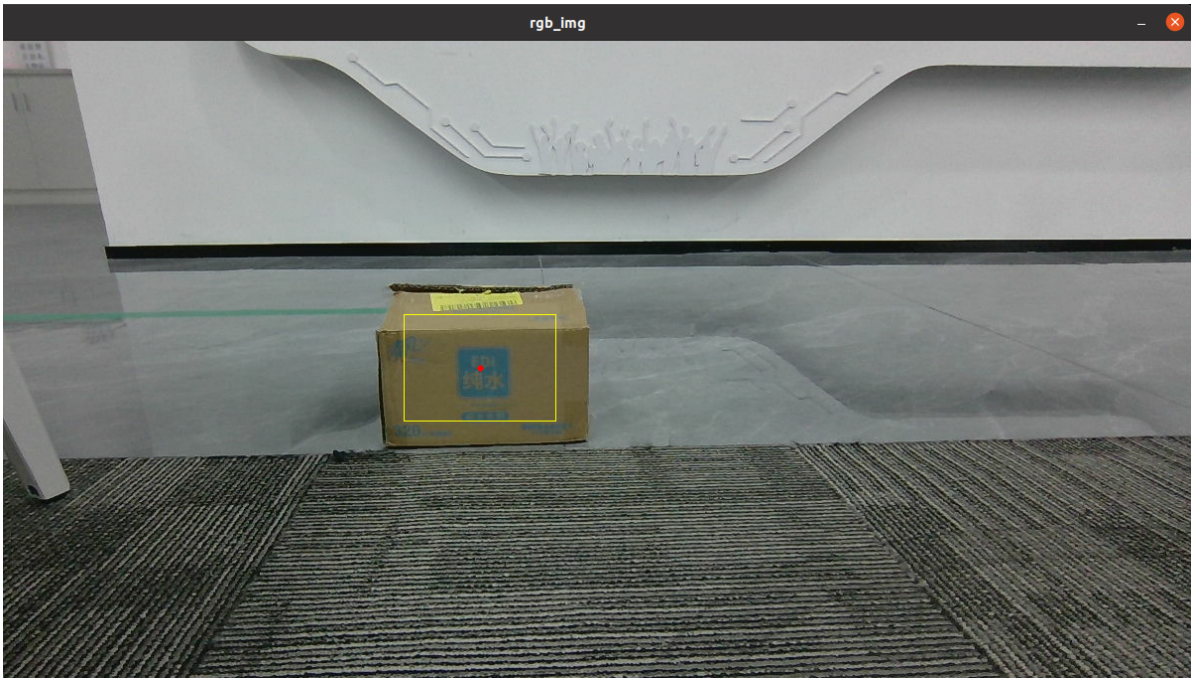
One-click start (robot side)

```
sudo supervisorctl start LaserServer #start/stop Switch radar service (indoor version)
roslaunch yahboom_navrobo_astra KCFTracker.launch
```

- The car has the front close-range and backward collision avoidance functions. It is necessary to ensure that the radar is started normally. If you run `rostopic echo /scan`

The print is empty and no data can be obtained, then the startup is abnormal. Please restart the radar service command.

After startup, enter the selection mode and select the location of the object with the mouse, as shown in the figure below. Release it to start recognition.



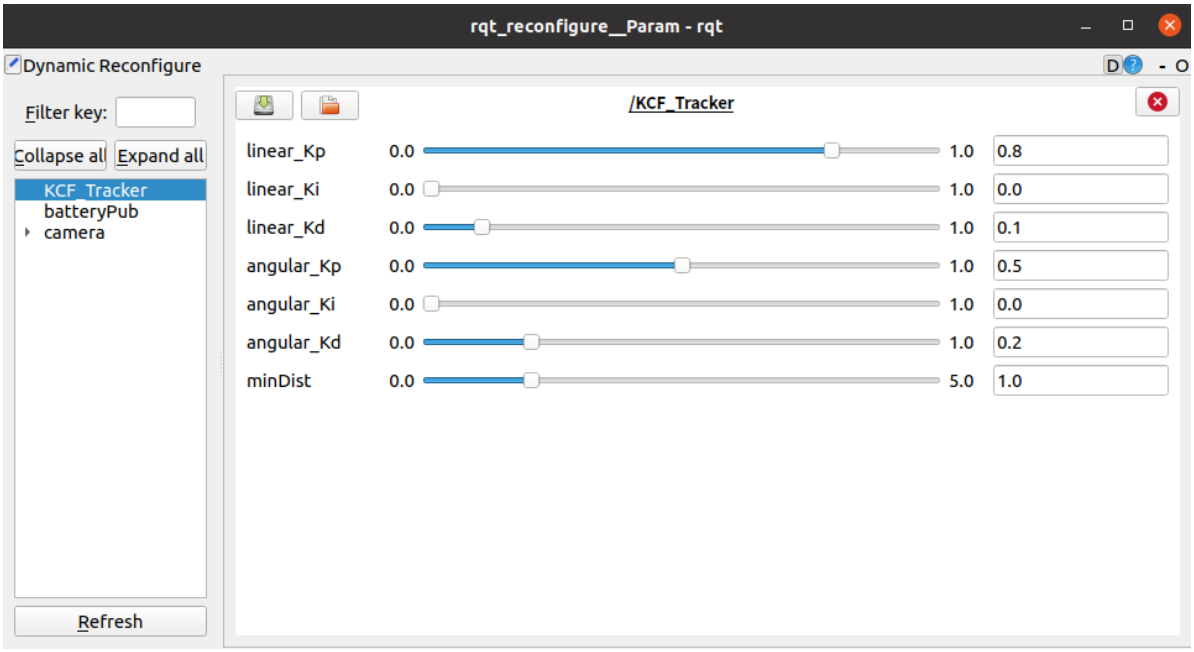
3.2.2, Keyboard control

[r]: Reset mode, you can use the mouse to select the area to identify the target.

[q]: Exit the program.

[Spacebar]: Target tracking; just move the target slowly when following, moving too fast will lose the target.

```
rosrun rqt_reconfigure rqt_reconfigure
```



Parameter analysis:

[linear_Kp], [linear_Ki], [linear_Kd]: Linear speed PID control during the car following process.

[angular_Kp], [angular_Ki], [angular_Kd]: Angular velocity PID control during the car following process.

[minDist]: Following distance, always maintain this distance.

- Parameter modification

When the parameters are adjusted to the optimal state, modify the corresponding parameters to the file, and no adjustment is required when using it again.

According to the optimal parameters of the [rqt_reconfigure] debugging tool, enter the [src] folder of the [yahboom_navrobo_astra] function package and modify the corresponding parameters of the [KCF_Tracker.cpp] file as shown below

```
ImageConverter::ImageConverter(ros::NodeHandle &n) {
KCFTracker tracker(HOG, FIXEDWINDOW, MULTISCALE, LAB);
float linear_KP=0.8;
float linear_KI=0.0;
float linear_KD=0.1;
float angular_KP=0.5;
float angular_KI=0.0;
float angular_KD=0.2;
```

The minDist parameter is modified in the [KCF_Tracker.h] file

```
float minDist = 1.0; ``` [rqt_reconfigure] Initial value modification of
debugging tool ```python gen.add("linear_Kp", double_t, 0, "Kp in PID", 0.8, 0,
1.0) gen.add("linear_Ki", double_t, 0, "Ki in PID", 0.0, 0, 1.0)
gen.add("linear_Kd", double_t, 0, "Kd in PID", 0.1, 0, 1.0)
gen.add("angular_Kp", double_t, 0, "Kp in PID", 0.5, 0, 1.0)
gen.add("angular_Ki", double_t, 0, "Ki in PID", 0.0, 0, 1.0)
gen.add("angular_Kd", double_t, 0, "Kd in PID", 0.2, 0, 1.0)
gen.add("minDist", double_t, 0, "minDist", 1.0, 0, 5.0)
exit(gen.generate(PACKAGE, "KCFTracker", "KCFTrackerPID"))
```

Enter the [cfg] folder of the [yahboom_navrobo_astra] function package and modify the initial values of the corresponding parameters in the [KCFTracker.cfg] file.

```
gen.add("linear_Kp", double_t, 0, "Kp in PID", 0.8, 0, 1.0)
```

Analyze the above example

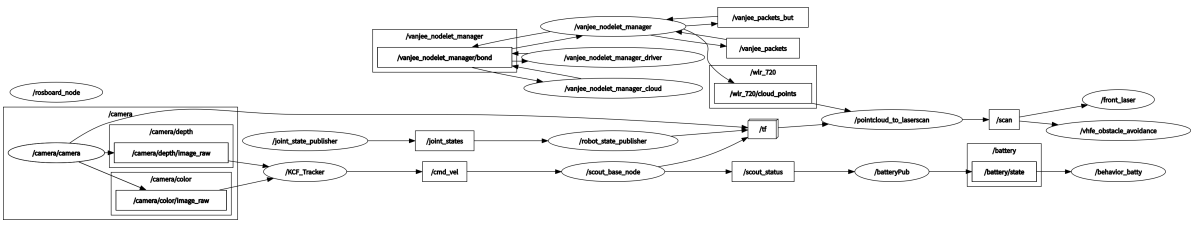
Parameter	Analysis	Corresponding parameter
name	Name of the parameter	"linear_Kp"
type	Parameter data type	double_t
level	A bit mask passed to the callback	0
description	A description parameter	"Kp in PID"
default	Initial value for node startup	0.8
min	Minimum value of parameter	0
max	Maximum value of parameter	1.0

Note: After the modification is completed, the environment must be recompiled and updated to be effective.

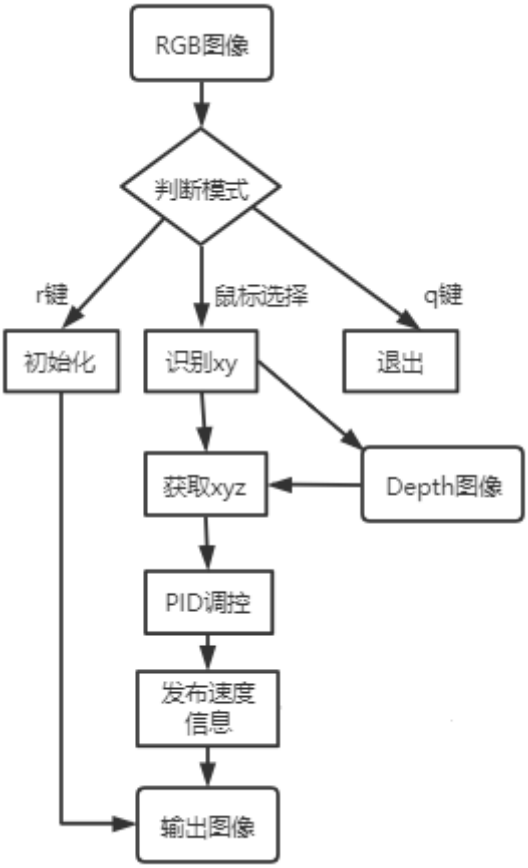
```
cd ~/YBAMR-COBOT-EDU-00001/
catkin build yahboom_navrobo_linefollw
source install/setup.bash
```

3.2.3, Node analysis

```
rqt_graph
```



[KCF_Tracker] Node analysis



- Subscribe to RGB color images
- Subscribe to depth images
- Publish car speed control instructions