4 Astra Object Tracking

4.1 Introduction

Feature pack: ~/yahboomcar_ws/src/yahboomcar_astra

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

- Object Tracking

Object tracking is all about 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 commonly studied under object tracking:

(1) Dense Optical flow(DOF): These algorithms help to 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, which tracks the positions of several feature points in the image.

(3) Kalman Filtering: A very popular signal processing algorithm based on prior motion information for predicting the position of moving objects. 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 maxima of the density function, and they are also used for tracking.

(5) Single object trackers: In this type of trackers, the first frame is marked with a rectangle to indicate the position of the object to be tracked. The object is then tracked in subsequent frames using a tracking algorithm. In most practical applications, these trackers are used together with object detectors.

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

- Comparison of Algorithms

algorithm	speed	precision	describe
BOOSTING	slow	Difference	The same machine learning algorithm behind Haar casades(AdaBoost), But it has been more than ten years since its birth, a veteran-level algorithm.
THOUSAND	slow	Difference	It is more accurate than BOOSTING, but the failure rate is higher.
KCF	quick	high	Faster than both BOOSTING and MIL, but underperformed with occlusion.
TLD	generally	generally	There are many false positives, and the phenomenon of follow-up is serious.
MEDIANFLOW	General+	generally	False positives are rare, and the model will fail for fast jumping or fast moving objects.
GOTURN	generally	generally	A deep learning based object detector that requires additional models to run.
MOVES	fastest	high-	The speed is really fast, but it is not as accurate as CSRT and KCF. You can choose it if you pursue speed.
CSRT	quick-	Highest	Slightly more accurate than KCF, but not as fast as KCF.

4.2 KCF object tracking

The full name of KCF is the Kernel Correlation Filter kernel correlation filtering algorithm. It was proposed by Joao F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista in 2014. After the algorithm came out, it was a sensation. This algorithm has a very dazzling performance in terms of tracking effect and tracking speed. Therefore, a large number of scholars have been researched on this algorithm, and the industry has gradually applied this algorithm to practical scenarios. this algorithm homepage that can be downloaded here, as well as some introductions. This article was posted on TPAMI in 2015, so you may see two versions, but no changes, all can be seen. The paper download address correlation filtering algorithm is a discriminative tracking, which mainly uses the given samples to train a discriminative classifier to determine whether the target or the surrounding background information is tracked. The rotation matrix is mainly used to collect the samples, and the fast Fourier transform is used to accelerate the calculation of the algorithm.

4.2.1 How to use

Note: [R2] of the remote controller has the function of [pause/on] for this gameplay.

One-click start(robot side)

```
roslaunch yahboomcar_astra KCFTracker.launch
```

After starting, enter the selection mode, use the mouse to select the location of the object, as shown in the figure below, release it to start recognition.

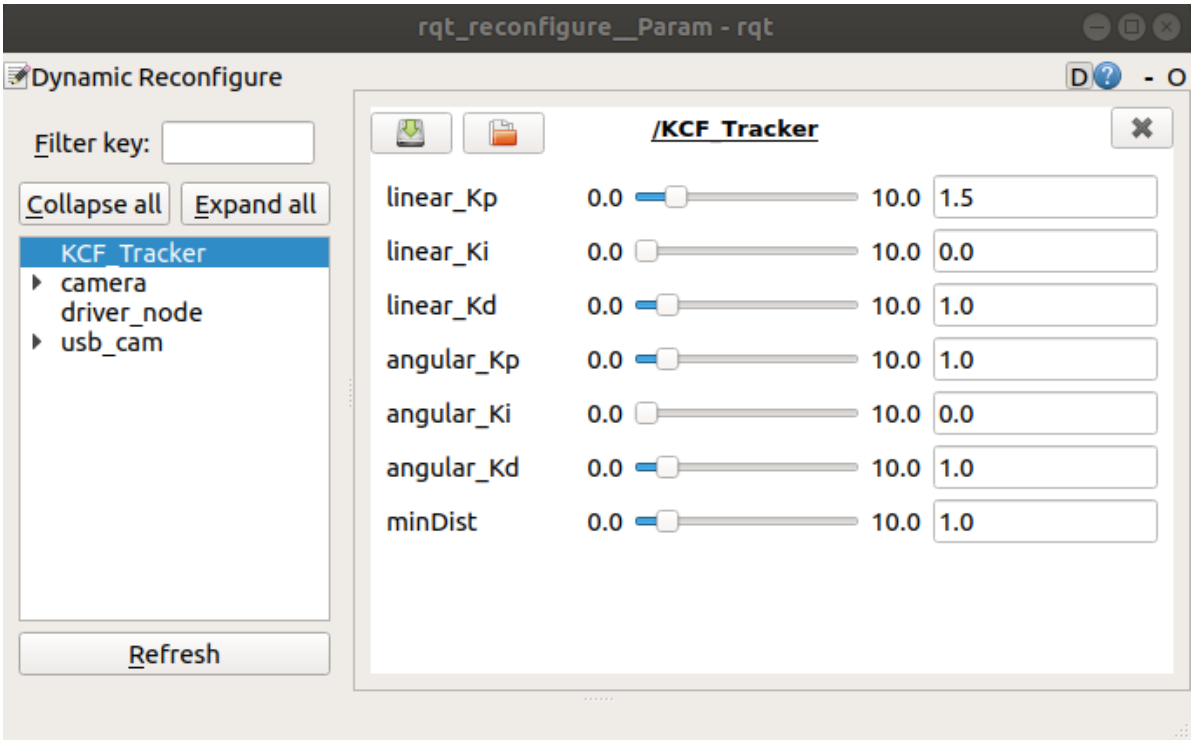
4.2.2 keyboard control

【r】: Reset mode, 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, otherwise the target will be lost if you move too fast.

```
rosrun rqt_reconfigure rqt_reconfigure
```



Parameter parsing:

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

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

[minDist]: Follow the distance and keep this distance all the time.

- parameter modification

When the parameters are adjusted to the optimal state, modify the corresponding parameters to the file, and there is no need to adjust them when using them again.

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

```
ImageConverter::ImageConverter(ros::NodeHandle  & n) {
    KCFTracker  tracker(HOG, FIXEDWINDOW, MULTISCALE, LAB);
    float  linear_KP = 0.9;
    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] Modify the initial value of the debugging tool

```
gen.add("linear_Kp", double_t, 0, "Kp in PID", 0.9, 0, 10.0)
gen.add("linear_Ki", double_t, 0, "Ki in PID", 0.0, 0, 10.0)
gen.add("linear_Kd", double_t, 0, "Kd in PID", 0.1, 0, 10.0)
gen.add("angular_Kp", double_t, 0, "Kp in PID", 0.5, 0, 10.0)
gen.add("angular_Ki", double_t, 0, "Ki in PID", 0.0, 0, 10.0)
gen.add("angular_Kd", double_t, 0, "Kd in PID", 0.2, 0, 10.0)
gen.add("minDist", double_t, 0, "minDist", 1.0, 0, 10.0)
exit(gen.generate(PACKAGE, "KCFTracker", "KCFTrackerPID"))
```

Enter the [cfg] folder of the [yahboomcar_astra] function package, and modify the initial values of the parameters corresponding to the [KCFTracker.cfg] file.

```
gen.add("linear_Kp", double_t, 0, "Kp in PID", 0.9, 0, 10.0)
```

Analysis of the above one as an example

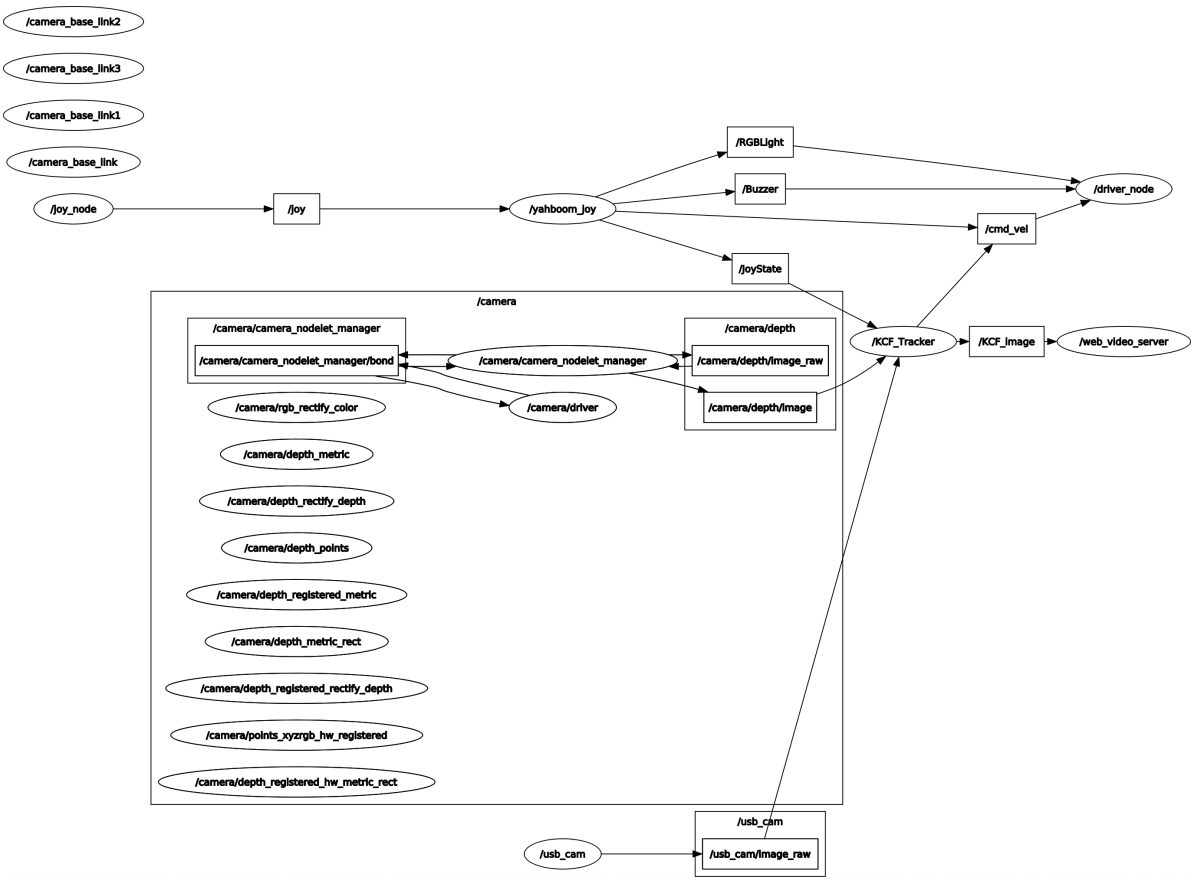
parameter	Parse	Corresponding parameters
name	the name of the parameter	"linear_Kp"
type	parameter data type	double_t
level	a bitmask passed to the callback	0
description	a description parameter	"Kp in PID"
default	Initial value for node startup	0.9
min	parameter minimum	0
max	parameter maximum	10.0

Note: After modification, the update environment must be recompiled to be effective.

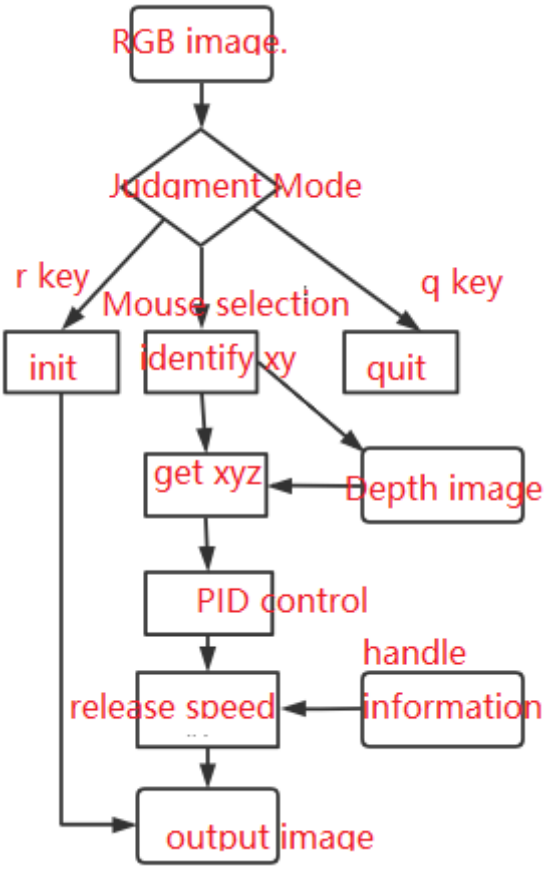
```
cd  ~/yahboomcar_ws
catkin_make
source  devel/setup.bash
```

4.2.3 Node analysis

```
rqt_graph
```



【KCF_Tracker】Node Analysis



- Subscribe to RGB Color Images
- Subscribe to depth images
- Subscribe to handle control information
- Issue trolley speed control command