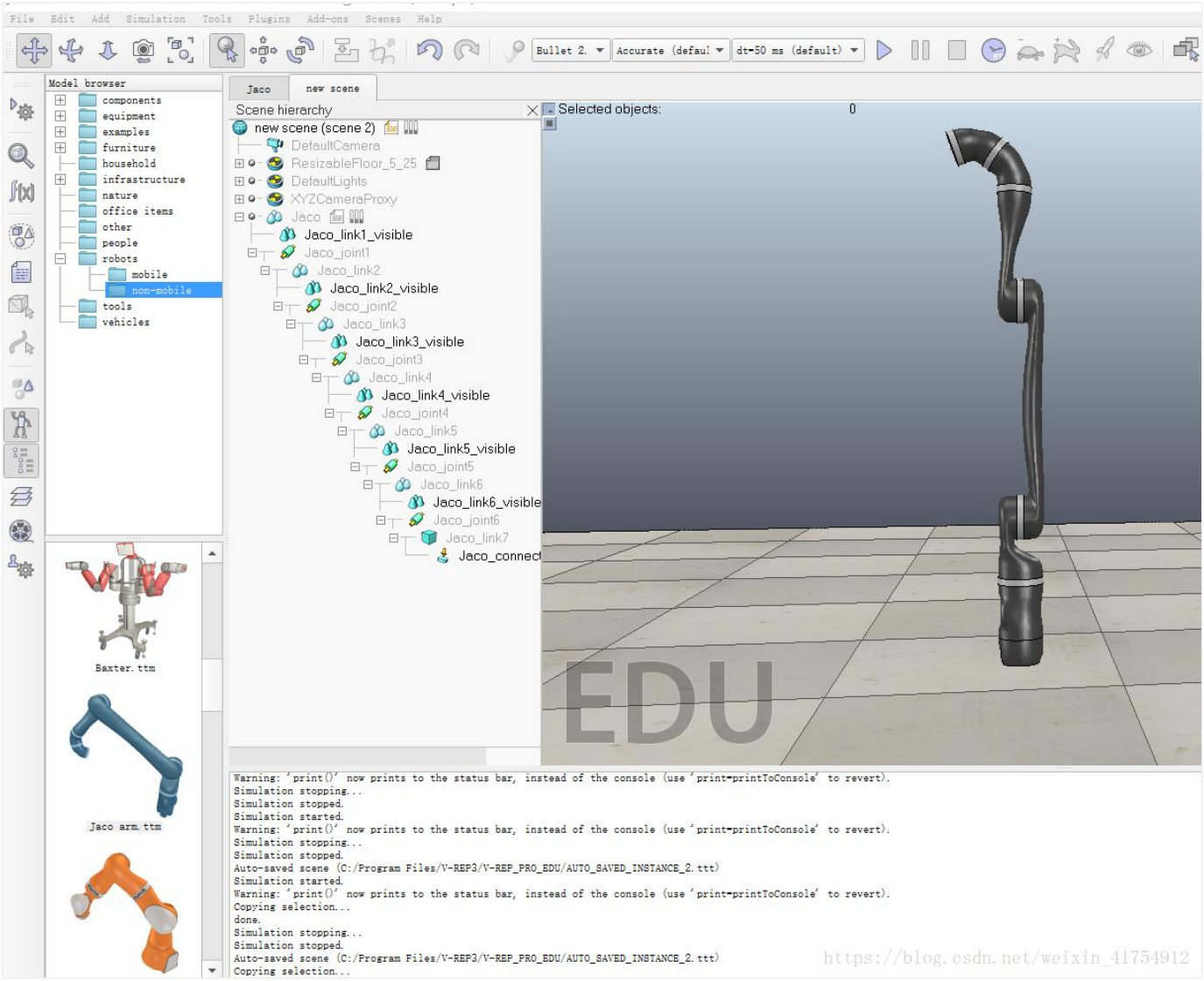
Python series of V-rep remote API control (1): Position control mode

In this section, we mainly introduce how to use remote API (Python) to control the movement of the robot arm, mainly introduce the position control mode.

```
1  In V-rep external control, we usually call the remote API side as the Client sid
```


1. Modeling

This part is not the focus of this section, so we directly select Jaco arm.ttm from [Model Brower]-> [robots]-> [non-mobile] on the left, drag it into a new scene, and save it.

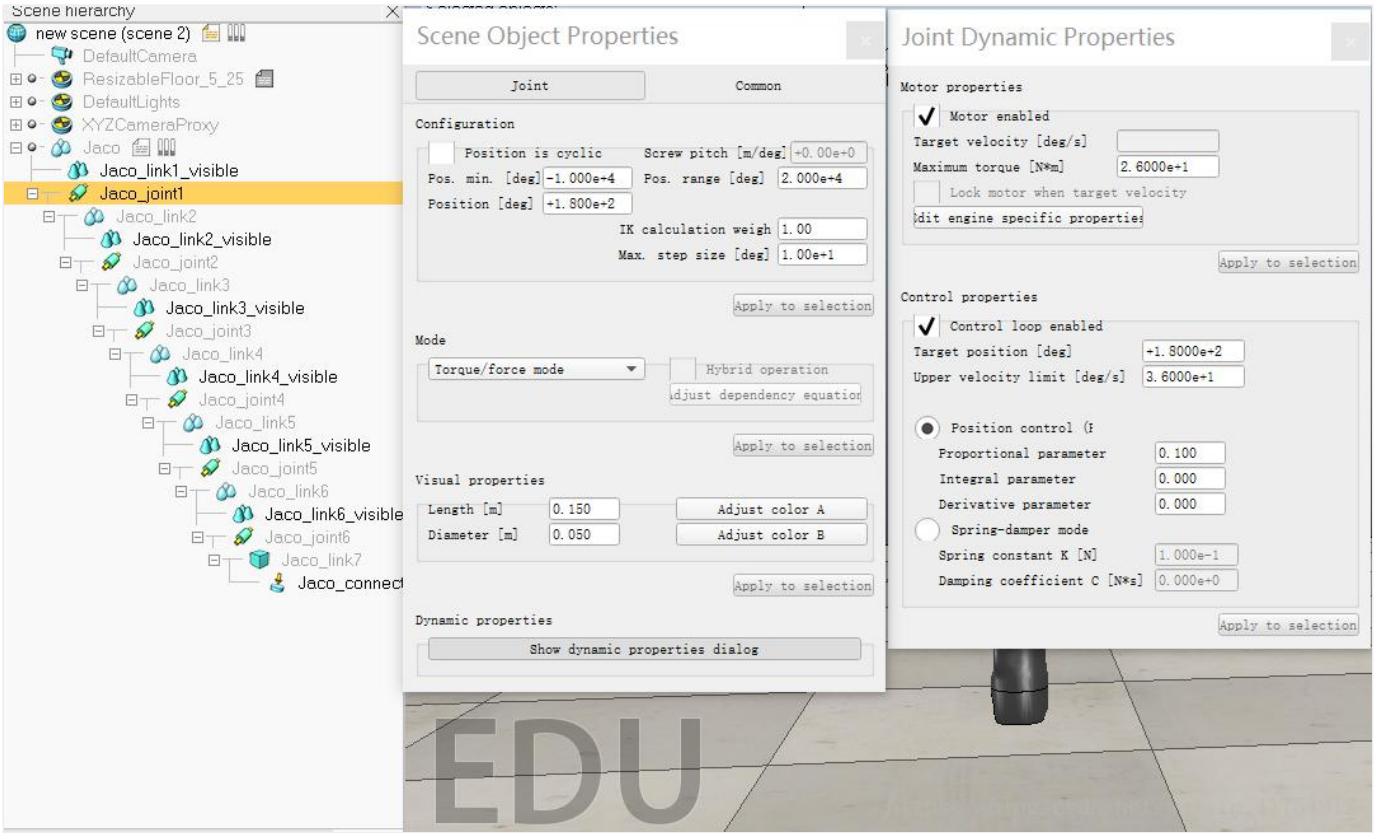


2. V-rep configuration

2.1 Joint configuration

Open the Object (that is, Jaco) in [Scene hiearchy], and you will find that there is a chain structure with Base, Joint, Link, etc. Today, we mainly configure Joint.

TOP



Double-click the icon in front of Jaco_joint *, the first pop-up window (Scene Object Properties) will pop up, you need to configure the following:

(1) Joint angle range [Pos.min, Pos.min + Pos.range]: I think it 's very clear to write this, for example, your joint angle rotation range is [- 90deg, 90deg], then you need to fill in -90 after Pos.min and 180 in Pos.range; (deg is to remind you that the unit here is °, not rad)

(2) The initial value of the joint angle Pos: All joints in Jaco default, the initial angle is 180 °, you can try to change the angle yourself to see how the configuration of Jaco changes;

(3) Mode: There are four control modes of Joint, and Torque / force mode is needed today;

Then enter Dynamic Properity: Click the button on the smallest side to enter the second pop-up window, Joint Dynamic Properties, which needs to be configured as follows:

(4) Motor enabled: select to enable the motor;

(5) Control loop enabled: select it to enter the default control;

(6) Target Position: target position, set to the initial value of the joint, because of external control, we will change it;

(7) Position control (PID): This is a form of default control, which we use today, and the little friends can come over and adjust PID parameters after completing this section , Look at the different parameters, the corresponding response. (To say one more thing: the robot arms of many manufacturers do not open the torque loop. The corresponding control is position control, suitable for beginners to adapt to the actual robot arm control.)

(8) Other parts do not need to be moved temporarily, pay attention to configure all joints! ! ! !

2.2 V-rep communication configuration

TOP

For external control, we usually use Non-threaded child script, so we need to configure this subscript.
We click on a "script" icon behind Object (Jaco), we will find its name is Thread child script, we delete it, and then add an associated child script (Non-threaded)

Open the script and you will find that there are four sub-functions

```
function sysCall_init()
function sysCall_actuation()
function sysCall_sensing()
function sysCall_cleanup()
```

According to the name, it is not difficult to guess, which are the configuration of initialization, actuator, sensor, clear and other operations. In order to give some friends who are not familiar with the Lua language better learning, we only configure communication in the initialization function, and the rest are done in the remote API.

Let's introduce this function: simRemoteApi.start, so that everyone will learn to understand Manual by themselves later.

simRemoteApi.start

Description	Starts a temporary remote API server service on the specified port. When started from a simulation script, the service will automatically end when the simulation finishes
Lua synopsis	number result=simRemoteApi.start(number portNumber,number maxPacketSize=1300,Boolean debug=false,Boolean preEnableTrigger=false)
Lua parameters	**portNumber**: port where to install the server service. Ports above 20000 are preferred. Negative port numbers can be specified in order to use shared memory, instead of socket communication. **maxPacketSize**: the maximum size of a socket send-packet. Make sure to keep the value at 1300, unless the client side has a different setting. **debug**: if true, a window will display the data traffic on that port. **preEnableTrigger**: if true, the server service will be pre-enabled for synchronous trigger signals from the client.
Lua return values	-1 if operation was not successful. In a future release, a more differentiated return value might be available

Function description: Start the temporary remote API server service on the specified port. When starting from the simulation script, the service will automatically end when the simulation is complete
Enter:
portNumber: The port where the server service is installed. Ports above 20000 have priority. In order to use shared memory, you can specify a negative port number instead of socket communication.
maxPacketSize: The maximum size of the packet sent by the socket. Make sure to keep the value at 1300 unless the client has different settings.
debug: If true, the window will show the data traffic on the port.
preEnableTrigger: If true, the server service will be pre-enabled for the synchronization trigger signal from the client.

TOP

> output:
> num result: -1 if the operation is unsuccessful. In future versions, there may be a more differentiated return value

[Youdao Dictionary Translator, serving friends who have headaches in English.] It can be roughly understood that this function usually enters some ports and other configurations to achieve temporary communication between the remote API and V-rep (the V-rep can only communicate when it is running !!! Remember, focus on, you need to test)

So, how to use it? Just copy and paste the following line into function sysCall_init (). Some people say that what you said so much, haha, just to make you play! !

```
repeat until (simRemoteApi.start(19999,1300,false,true)~=-1)
```

This completes the configuration of the V-rep side. The following describes how to configure the Python side.

3. API side configuration

3.1 Python configuration

The author uses Python 3.6 and the compiler is PyCharm. If you want to know how to install Python and PyCharm, go out and turn right, and read the blog of the author's software installation section.

3.2 Import of function library

TOP

Python client

To use the remote API functionality in your Python script, you will need following 3 items:

- vrep.py
- vrepConst.py
- remoteApi.dll, remoteApi.dylib or remoteApi.so (depending on your target platform)

Above files are located in V-REP's installation directory, under *programming/remoteApiBindings/python*. You might have to build the remoteApi shared library yourself (using *remoteApiSharedLib.vcproj* or *makefile*) if not already built. In that case, make sure you have defined **NON_MATLAB_PARSING** and **MAX_EXT_API_CONNECTIONS=255** (and optionally **DO_NOT_USE_SHARED_MEMORY**) as a preprocessor definition.

Once you have above elements in a directory known to Python, call *import vrep* to load the library. To enable the remote API on the client side (i.e. your application), call *vrep.simxStart*. See the *simpleTest.py* script in the *programming/remoteApiBindings/python* directory for an example. This page lists and describes all supported Python remote API functions. V-REP remote API functions can easily be recognized from their "simx"-prefix.

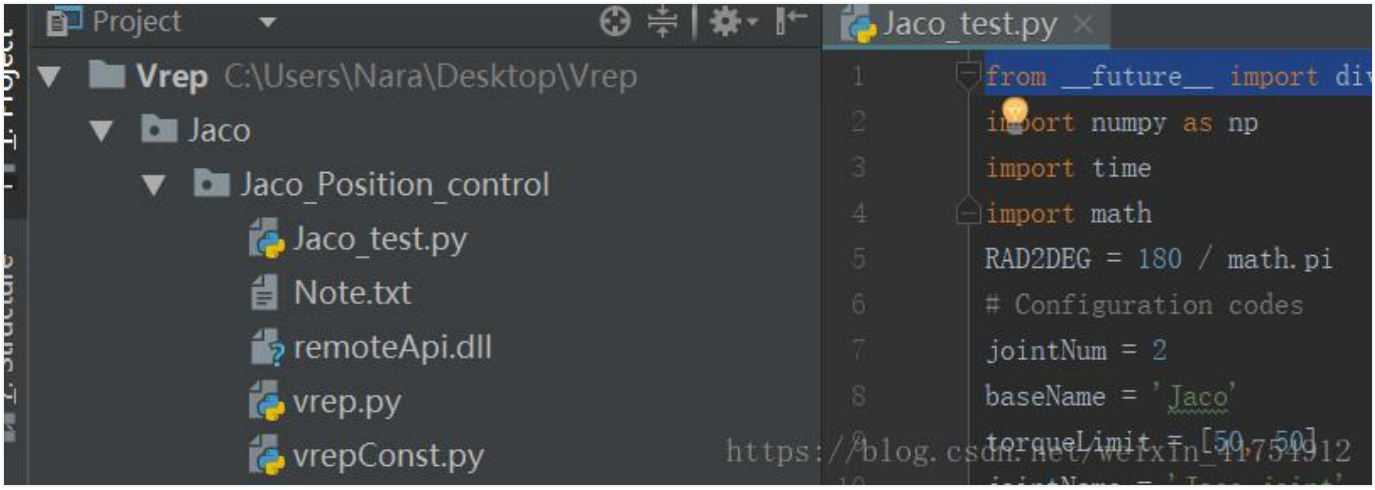
To put it simply, you need to configure several files, just like pip install numpy, install some external libraries, add interface related files in Python, these files can be installed in Vrep Found in the folder, including:

> vrep.py
> vrepConst.py
> remoteApi.dll(win) remoteApi.dylib(mac) remoteApi.so(linux)

Vrep installation folder-> programming-> remoteApiBindings->
(1)->python->python
(1)->lib->lib->(32/64 Bit)

Copy these files to the Project folder defined by PyCharm, which also saves trouble, as shown in the figure.

TOP



4. Python programming

4.1 Steps of simulation

> 1. Setting (setting introduces some library functions, parameters, etc.)
> 2. Init (close other connections, communication detection)
> 3. Configuration (configuration handle, setting synchronization mode, etc.)
> 4. Simulation (Set Simulation Parameters)
> 5. CoreControl (write planning and control in the simulation loop)

Step1. Setting

```
from __future__ import division
import numpy as np
import math
import vrep

RAD2EDG = 180 / math.pi    # Constant, radians to degrees
tstep = 0.005              # Define the simulation step
# Configure joint information
jointNum = 6
baseName = 'Jaco'
jointName = 'Jaco_joint'
```

Step2. Init

```
print('Program started')
# Close potential connections
```

TOP

```
vrep.simxFinish(-1)
# Check every 0.2s until connected to V-rep
while True:
    clientID = vrep.simxStart('127.0.0.1', 19999, True, True, 5000, 5)
    if clientID > -1:
        break
    else:
        time.sleep(0.2)
        print("Failed connecting to remote API server!")
print("Connection success!")
```

Step3. Configuration

```
# Set the simulation step size in order to keep the API and V-rep synchronized
vrep.simxSetFloatingParameter(clientID, vrep.sim_floatparam_simulation_time_step
# Then turn on sync mode
vrep.simxSynchronous(clientID, True)
vrep.simxStartSimulation(clientID, vrep.simx_opmode_oneshot)

# Then read the handles of Base and Joint
jointHandle = np.zeros((jointNum,), dtype=np.int) # Note that it is an integer
for i in range(jointNum):
    _, returnHandle = vrep.simxGetObjectHandle(clientID, jointName + str(i+1), v
    jointHandle[i] = returnHandle

_, baseHandle = vrep.simxGetObjectHandle(clientID, baseName, vrep.simx_opmode_bl

print('Handles available!')

# Then read the initial value of the joint for the first time, in the form of st
jointConfig = np.zeros((jointNum,))
for i in range(jointNum):
     _, jpos = vrep.simxGetJointPosition(clientID, jointHandle[i], vrep.simx_opm
     jointConfig[i] = jpos
```

Step4: Simulation

```
lastCmdTime=vrep.simxGetLastCmdTime(clientID)  # Record the current time
vrep.simxSynchronousTrigger(clientID)  # Let the simulation take a step
# Start simulation
while vrep.simxGetConnectionId(clientID) != -1 :
    currCmdTime=vrep.simxGetLastCmdTime(clientID)  # Record the current time
    dt = currCmdTime - lastCmdTime # Record time interval for control
    # ***
    # ***
```

TOP

```
    # ***
    lastCmdTime=currCmdTime    # Record the current time
    vrep.simxSynchronousTrigger(clientID)  # Take the next step
    vrep.simxGetPingTime(clientID)    # Make the simulation step complete
```

Note: in***You can also write the code of the CoreControl part inserted at

Step5. CoreControl

```
        # Read the current state value, and then use buffer to read
        for i in range(jointNum):
            _, jpos = vrep.simxGetJointPosition(clientID, jointHandle[i], vrep.s
            print(round(jpos * RAD2DEG, 2))
            jointConfig[i] = jpos

        # Control commands need to be in simultaneous mode, so the communication
        vrep.simxPauseCommunication(clientID, True)
        for i in range(jointNum):
            vrep.simxSetJointTargetPosition(clientID, jointHandle[i], 120/RAD2DE
        vrep.simxPauseCommunication(clientID, False)
```

Well, the writing is over; Note: The expected position of the author to control all joints is 120 °, of course, you can try more complicated control commands, according to the above template.

5. Simulation

First run the simulation on the V-rep side, then run the code on the Python side, you can find that all the joints are rotated to the 120 ° position.

I wish you all a smooth study!

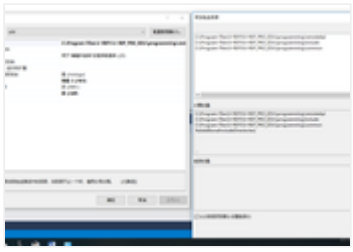
Reference

[1]. V-rep user manual
[2]. https://blog.csdn.net/huangdianye/article/details/80628932

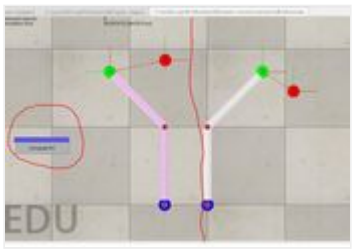
Intelligent Recommendation

Introduction to v-rep-VS2015 c++ control car

TOP



Introduction usevs2015IDEwritewin32Console, via port andv-repCommunication, control car movement The issue to note is that currently bloggers only find 32-bit static librariesremoteApi.lib, Because th...



Robot simulation software V-REP learning record Robotic arm control 0-1

Robot arm control part 1 Preview--case demo1 - singleIkGroupWithSingleIkElement-undamped (single joint inverse kinematics-no damping) demo2 - singleIkGroupWithSingleIkElement-damped (single joint invers...

In-depth analysis of the application of V-REP Remote API (MATLAB)

OS: Win10 x64 V-REP: V-REP PRO EDU 3.5.0 MATLAB: 2016b Let's talk about MATLAB in V-REPRemote APIApplications. If you only have a basic understanding of V-REP and are not familiar with V-REP's remote ...

V-REP regular API (API used in lua script)---(1)

Article Directory V-REP API framework Regular API Regular API Regular API function list (by category) Regular API function list (by category) File operations General object handle retrieval General fu...

[V-REP self-study notes (3)] Use code to control the robot

Lead In this series of V-REP self-study notes, we set a small goal to complete a Demo. Use the official YouBot robot model provided by KUKA to experiment with the robot's perception and control ...

TOP

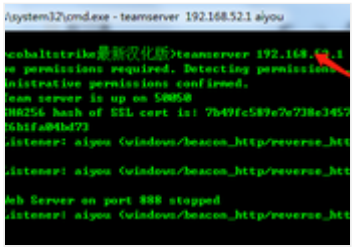
More Recommendation

[V-REP self-study notes (4)] keyboard cont rol YouBot robot

Lead In this series of V-REP self-study notes, we set a small goal to complete a Demo. Use the official YouBot robot model provided by KUKA to experiment with the robot's perception and control ...

Keyboard of V-REP simulation control robo t arm end movement

Keyboard of V-REP simulation control robot arm end movement The keyboard controls the end of the end of the robot arm is to set the robot arm internatology Target, and the end of the robot is followed...

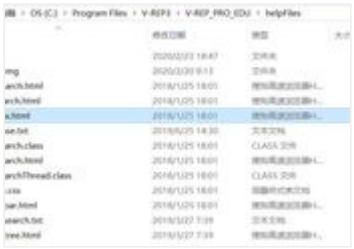


Cobaltstrike series (a) - Remote control

The day before yesterday in the sun and father, my father suddenly said something: how does it feel, social good break it, then it slowly if too tired, I hold you to it. Hearing this, when I suddenly ...

V-REP study notes (1)

TOP



V-REP (Virtual Robot Experimentation Platform) is a robot simulation software in the "Swiss Army knife." In 2019 this long winter, I began to gradually explore V-REP. When I reviewed books o...

First acquaintance with V-REP (1)

** First acquaintance with V-REP (1) ** 2020 is a special year. Due to the impact of new coronary pneumonia, there has been no resumption of school, and it is completely online at home. Because of the...

Related Posts

- V-rep------python remote control Baxter
- V-REP Simulation and Python Remote Control
- Introduction to v-rep (1) remote api python
- [V-REP self-study notes (7)] Matlab/Python remote control
- V-REP Remote API
- v-rep uses ompl planning library and Python control to complete the capture
- V-REP Simulation Python Control Arm Arm and RG2 Claw
- How to use V-rep remote API module
- V-REP Tutorial (9): Motor Control

TOP

- v-rep entry -matlab control vehicle movement

Popular Posts

- JavaFX learning example 6
- jQuery or selector?
- LeetCode 011 container with the most water
- Detailed C language structure
- Elementary Data Structure
- C # achieve the specified cell operates EXCEL
- Small trap on xxxxxxRepository.search () method of a page
- Python implements convolution calculation, smoothing filter, Gaussian filter, Sobel operator, Prewitt operator
- [Solution] Luogu P2661 [NOIP2015] information transmission and collection
- Study notes for the MDSeq-R package (direct paste code)

TOP

Recommended Posts

- Perfect number
- Using tags in Cocos2d-x Lua
- The Java Evolution of the Eclipse Collection
- Use Linux Raspberry Pi to read JY61 serial port data
- About adding a final in the code set to facilitate the reasonable allocation of application memory
- --- HTML front-end
- MyTools
- Explain Angular's custom instructions in detail
- AcWing 1012. Sister cities
- Three advantages of network monitoring

Related Tags

V-rep

baxter

python

Smart car

Machine learning

simulation

TOP

v-rep

remote api

artificial intelligence

Programming language