Fruit classification with image recognition & robot arm

I. Experimental Object

The purpose of this experiment was to develop an automatic object classification system based on image recognition and robotic arm control. Capture images in real time through the camera, identify the fruit on the object, and control the robotic arm to grab and classify the object. Finally, automatic object classification is realized to improve production efficiency.

II. Experimental Content

This experiment is a comprehensive experiment that combines the training recognition of the Nanodet with the grasping of the robotic arm, and the recognition is carried out by the visual model Nanodet, and the robotic arm classifies the fruits according to the recognition results. In this process, the main thing is to master the whole control logic and the principles and methods related to the teaching of the robotic arm and vision.

At the same time, this experiment is an expansive experiment, classify the identified fruits, expand thinking, and consolidate the knowledge learned before, combining machine vision with robotic arms can do a lot of interesting experiments, and students can try different ways to play according to their interests.

Experimental steps:

1.Use a camera to capture live images.

2.Identify fruit on objects based on Nanodet.

3.The recognition results are sent to the robotic arm control system.

4.The robotic arm grabs the object according to the recognition result and sorts it and places it in the specified position.

III. Experimental Environment

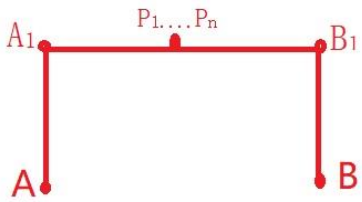
Experimental equipment	AI Comprehensive Experiment Box
Operating system	Linux
Experimental accessories	2D camera, Robotic arm.

IV. Experimental Principle

The image is acquired by a 2D camera, and the object is recognized based on the Nanodet, and the pixel center point of the object is located at the same time. Through visual calibration, the pixel center point of the object is converted into the world coordinates of the robotic arm, and the robotic arm grabs the object and classifies it according to the recognized fruits.

First, the position of the classification is defined in advance according to your own needs, and the position can be modified arbitrarily according to the experimental requirements. In the process of teaching the robotic arm, the movement path planning should be reasonable according to the situation, and the movement path of the robotic arm should be a "door-shaped" movement. That is, the robotic arm grabs the object from point A and places it at point B, and the motion trajectory should not only teach two points, point A and point B, so that the robotic arm will collide during the movement and interfere with other points.

The correct teaching step should be to teach an A1 from the position of the teaching point A directly above, teach a B1 directly above the teaching point B, the transition point between A1 and B1 according to the situation, the principle of the transition point is not to interfere with other teaching points, as shown in the figure below.



The visual calibration(calibration.py), Nanodet recognition(detect_main.py), robot control (Five_Robot_Control.py), and robot kinematics forward and inverse (Five_Robot_kinematics.py) solution program files required in the experiment have been classified and encapsulated, and only need to be called as needed when used. The purpose of this encapsulation is to facilitate maintenance, modification, and clearer thinking, and the functions of each piece are separated independently.

The main program file "Robotic_fruit_sorting.py" is built as a shelf, which connects each function block in parallel and fills the business logic to make it a complete host computer function. The detailed business process of the entire framework is as follows.

The first is to capture photos through a camera. The photo acquisition program uses a timer (triggered at certain intervals), because the acquisition of images is always required, and the process operation of the entire interface is a single-threaded mode, so you cannot wait until a photo is processed before the next one is collected, such a program is very difficult to use, and it does not meet the requirements. Under normal circumstances, images need to be captured at any time, so a multi-threaded approach is required.

After the image acquisition is completed, the image needs to be processed according to the business logic of the interface, the business logic of the interface is that the camera is continuously displayed when the program is opened, and a photo processing is collected when the detection is turned on, and the processing result is displayed in the interface.

In this way, it is necessary to turn on the timer when the interface class is initialized, continuously collect and display the image, the detection button of the interface class, control the collection of a frame of image, and call the Nanodet recognition class function to process and recognize the pixel value of the center point of the fruit type and fruit box. The pixel value of the center point is obtained and the visual calibration function is called to obtain the calibration data, and the world coordinates are obtained after obtaining, and then the robot kinematics forward and reverse solution class function is called to obtain the joint coordinates of the robot target, and the joint coordinates are obtained and the robot control function is called, and the joint parameters are input to control the robot to grasp the target. After grabbing the target, the robotic arm control function is invoked to recognize the characters and move to the target point according to the Nanodet.

This is the business logic of the whole program, a fruit classification experiment from the acquisition of images→ recognition of images→ visual calibration→

kinematic inverse solution→ robot control grabbing→ fruit classification of the whole process ends.

The flowing of the experimental principle:

(1) Recognition

1.Image capture: Capture live images via the camera.

2.Digit recognition: The fruit in the image is identified based on the Nanodet model.

3.Hand-eye calibration: The pixel coordinates in the image are converted to the actual world coordinates to ensure that the manipulator can accurately locate the object.

(2) Robotic arm control system

1.Kinematic calculation: According to the world coordinates of the object, calculate the joint angle of the robotic arm, so that the end of the robotic arm reaches the target position.

2. Torque control: Turn on the torque of the robotic arm so that it can perform grab and place operations.

3. Motion control: control the robotic arm to move according to the calculated joint angle to complete the grasping and classification of objects.

V. Experimental Procedure

5.1 Running Jupyter lab

(1) Open the folder named "Experiments", click the right mouse button in the blank. Select "Open Terminal", and enter "jupyter lab" in the terminal interface.

(2) In the folder name of the Jupyter Lab programming interface, select "8.基于视觉的机器人应用" in "Source Program(源程序)" to open the "8.基于视觉的机械臂水果分类" folder;

(3) On the startup page of the Jupyter Lab programming interface, select Python3 under the notebook menu to enter the program editor.

(4) Right-click the program file with the ".ipynb "endnote and select" Rename "to name the program the name of the experiment performed.

5.2 Import library files and methods

Import the libraries and modules needed for the experiment.cv2 is used for computer vision related operations such as reading and displaying images, time is used for timing control, and threading is used for multithreading. The remaining custom modules are used for object recognition, camera calibration, manipulator kinematics calculation, and manipulator control, respectively.

***The example code is as follows: ***

Creating a new program file "基于视觉的机械臂水果分类.ipynb"

```
# region 库文件(Library files)
import cv2
import time
import threading
from detect_main import *
from calibration import *  # 导入校准模块(Import the calibration module)
from Five_Robot_kinematics import *  # 导入机械臂运动学模块(Introduced the Robotic Arm Kinematics Module)
from Five_Robot_Control import *  # 导入机械臂控制模块(Introduced the robot arm control module)
# endregion
```

5.3 Define global variables and instantiate objects

Define global variables for storing and sharing data, such as the current image, object count, detection results, and so on. Instantiate objects that control robotic arms, recognize objects, calibrate cameras, and calculate kinematics for invocation in subsequent functions.

The example code is as follows:

Proceed with the program file "基于视觉的机械臂水果分类.ipynb"

```
# region 全局变量(Global variables)
global shape_count_circles  # 圆形计数(Count the number of circles)
shap_count_circles = 0
global shape_count_square  # 正方形计数(Count the number of squares)
shape_count_square = 0
global result  # 检测结果(Test results)
result = []
global bool_IsSend  # 发送数据标志(Send data flags)
bool_IsSend = False
```

```
global clip_open_degree   # 夹爪每次放置物体时张开夹爪度数，防止夹爪张开过大碰撞到其它(The gripper opens the gripper degree every time the object is placed, and attention is paid to prevent the gripper from opening too much and colliding with other objects.)
clip_open_degree = 40
global clip_close_degree   # 夹爪关闭时角度(Angle when the gripper is closed)
clip_close_degree = 75
# endregion
# region  实例化对象(Instantiate the object)
robot_control = Blinx_Five_Robot_Control()   # 机械臂控制实例(Example of robotic arm control)
object_recognition = Blinx_my_nanodet() #  水果识别实例(Examples of fruit identification)
calibration = Blinx_Cam_Calibration()   # 摄像头校准实例(Example of camera calibration)
five_robot_kinematics = Blinx_Five_Robot_kinematics()   # 机械臂运动学实例(Example of robotic arm kinematics)
# endregion
#  定义全局变量 image，用于存储当前帧图像
# Define global variable ‘image’. Used to store the image of the current frame
global image
```

5.4 Send data thread function

Run in a separate thread, periodically checking bool_IsSend flags. If the flag is True, call the blinx_robot_pick function to perform the gripping operation of the robotic arm, and then reset the bool_IsSend flag to False. After processing, clear the list of result.

The example code is as follows:

Proceed with the program file "基于视觉的机械臂水果分类.ipynb"

```
#  发送数据线程函数
# Send data thread function
def blinx_send_data():
    try:
        while True:
            time.sleep(0.1)
            global bool_IsSend
            if bool_IsSend:
                bool_IsSend = False
                global result
```

```
                    if result:
                        # 调用机械臂抓取函数
                        # Call the robotic arm gripping function
                        blinx_robot_pick(int(result[0]), int(result[1]))
                        print("数据已发送到机械臂(The data has been sent to
the robotic arm)")
                        result = []
                    else:
                        print('图像识别错误(Image recognition errors)')
        except Exception as e:
            print("数据发送线程异常(The data sending thread is abnormal)： ",
e)
```

5.5 The grasping function of Robotic Arm

Convert the detected object pixel coordinates to world coordinates, control the robotic arm by calculating joint angles. Move the robotic arm to the target position, perform grabbing and releasing actions, and call the blinx_object_classify function for object classification.

***The example code is as follows: ***

Proceed with the program file "基于视觉的机械臂水果分类.ipynb"

```
# 机械臂抓取函数
# Mechanical arm grabbing function.
def blinx_robot_pick(pixel_x, pixel_y):
    # 通过校准将像素坐标转换为世界坐标
    # Convert pixel coordinates to world coordinates by calibration
    point_x, point_y = calibration.blinx_calibration(pixel_x, pixel_y)
    print("世界坐标(World coordinates):", point_x, point_y)
    # 计算关节角度
    # Calculate joint angles
    arr1, arr2, arr3, arr4 = five_robot_kinematics.arr(point_x, point_y, 0)
    print("关节角度(Joint angles):", arr1, arr2, arr3, arr4)
    # 开启机械臂
    # Turn on the robotic arm
    robot_control.blinx_bus_servo_niuju_on(0xfe)
    time.sleep(0.5)
    # 控制机械臂运动
    # Control the movement of the robotic arm
    robot_control.blinx_bus_servo_all(30, 58, 170, 217, 0, 1000)
    time.sleep(2)
    robot_control.blinx_bus_servo_all(arr1, arr2, arr3, arr4, 0, 1000)
    time.sleep(2)
```

```
    robot_control.blinx_bus_servo_all(arr1, arr2, arr3, arr4, clip_close_degree,
1000)
    time.sleep(2)
    robot_control.blinx_bus_servo_all(30, 74, 172, 198, clip_close_degree, 1000)
    time.sleep(2)
    robot_control.blinx_bus_servo_all(200, 74, 172, 198, clip_close_degree, 1000)
    time.sleep(2)
    global result
    # 调用物体分类函数
    # Call the object classification function
    blinx_object_classify(result[2])
    robot_control.blinx_bus_servo_all(200, 74, 172, 198, 0, 1000)
    time.sleep(2)
    robot_control.blinx_bus_servo_all(30, 58, 170, 217, 0, 1000)
```

5.6 Function for object classification

Depending on the fruit on the object, the object is placed in a predefined position. Different fruit objects correspond to different placement points.

The example code is as follows:

Proceed with the program file "基于视觉的机械臂水果分类.ipynb"

```
# 物体分类函数
# Object classification function
def blinx_object_classify(fruit):
    global clip_open_degree
    global clip_close_degree
    # 定义四个放置点的坐标和角度
    # Define the coordinates and angles of the four drop points
    point1_place = [213, 96, 193, 204, clip_close_degree, 1000]
    point2_place = [192, 96, 193, 204, clip_close_degree, 1000]
    point3_place = [220, 47, 210, 227, clip_close_degree, 1000]
    point4_place = [185, 47, 210, 227, clip_close_degree, 1000]
    # 根据水果分类
    # Classification according to fruits
    if fruit == "banana":
        robot_control.blinx_bus_servo_all(point1_place[0], point1_place[1],
point1_place[2], point1_place[3],
                                          point1_place[4], point1_place[5])
        time.sleep(1)
        robot_control.blinx_bus_servo_all(point1_place[0], point1_place[1],
point1_place[2], point1_place[3],
                                          clip_close_degree - clip_open_degree,
point1_place[5])
```

```
            time.sleep(1)
            robot_control.blinx_bus_servo_all(point1_place[0], point1_place[1],
point1_place[2], 207,
                                              clip_close_degree - clip_open_degree,
point1_place[5])
            time.sleep(1)
        elif fruit == "tomato":
            robot_control.blinx_bus_servo_all(point2_place[0], point2_place[1],
point2_place[2], point2_place[3],
                                              point2_place[4], point2_place[5])
            time.sleep(1)
            robot_control.blinx_bus_servo_all(point2_place[0], point2_place[1],
point2_place[2], point2_place[3],
                                              clip_close_degree - clip_open_degree,
point2_place[5])
            time.sleep(1)
            robot_control.blinx_bus_servo_all(point2_place[0], point2_place[1],
point2_place[2], 207,
                                              clip_close_degree - clip_open_degree,
point2_place[5])
            time.sleep(1)
        elif fruit == "watermelon":
            robot_control.blinx_bus_servo_all(point3_place[0], point3_place[1],
point3_place[2], point3_place[3],
                                              point3_place[4], point3_place[5])
            time.sleep(1)
            robot_control.blinx_bus_servo_all(point3_place[0], point3_place[1],
point3_place[2], point3_place[3],
                                              clip_close_degree - clip_open_degree,
point3_place[5])
            time.sleep(1)
            robot_control.blinx_bus_servo_all(point3_place[0], point3_place[1],
point3_place[2], 207,
                                              clip_close_degree - clip_open_degree,
point3_place[5])
            time.sleep(1)
        elif fruit == "cucumber":
            robot_control.blinx_bus_servo_all(point4_place[0], point4_place[1],
point4_place[2], point4_place[3],
                                              point4_place[4], point4_place[5])
            time.sleep(1)
            robot_control.blinx_bus_servo_all(point4_place[0], point4_place[1],
point4_place[2], point4_place[3],
                                              clip_close_degree - clip_open_degree,
```

```
point4_place[5])
            time.sleep(1)
            robot_control.blinx_bus_servo_all(point4_place[0], point4_place[1],
point4_place[2], 207,
                                              clip_close_degree - clip_open_degree,
point4_place[5])
            time.sleep(1)
        else:
            print("未知的水果(Unknown fruit)")
```

5.7 Function to enable the camera

Turn on the camera with the specified number and check if it turns on successfully. If it is not opened successfully, an exception is thrown.

The example code is as follows:

Proceed with the program file "基于视觉的机械臂水果分类.ipynb"

```
# 打开摄像头函数
# Open camera function
def blinx_open_camera(cam_num):
    cap = cv2.VideoCapture(cam_num)  # 打开摄像头(Open camera)
    if not cap.isOpened():  # 检查摄像头是否成功打开(Check if the camera
turns on successfully)
        raise Exception("请检查相机与电脑连接是否正确(Please check that
the camera is connected to the computer correctly)")
    return cap  # 返回摄像头对象(Return to the camera object)
```

5.8 Processing identifying objects or fruits

Perform object recognition on the incoming image, obtain the recognition result, and display the recognized image. If the recognition is successful, set the flag "bool_IsSend" to True, update the variable "result", and wait for the thread function "blinx_send_data" to process it.

The example code is as follows:

Proceed with the program file "基于视觉的机械臂水果分类.ipynb"

```
# 开始进行物体水果识别，并显示识别后的图像
# Begin object fruit recognition and display the recognized image
def blinx_start_detection(image):
    try:
        # 调用物体水果识别函数
        # Call the object fruit recognition function
        img, identify_result = object_recognition.blinx_detect(image)  # 水果
```

```
识别(Fruit recognition)
            cv2.imshow("Camera", img)   # 显示识别后的图像(The image after
recognition is displayed)
            global bool_IsSend, result
            # 是否打开机械臂抓取
            # Whether to turn on the robotic arm to grasp
            bool_IsSend = True
            result = identify_result
            cv2.waitKey(0)
        except Exception as e:
            print("数据获取失败(Data fetching failed)：", e)
```

5.9 Defining the main program

The main program cycles through the camera image and displays it. If the 'd' key is pressed, the blinx_start_detection is called for object recognition; If the 'Q' key is pressed, the cycle is exited. Release the camera and close all OpenCV windows when the program ends.

The example code is as follows:

Proceed with the program file "基于视觉的机械臂水果分类.ipynb"

```
# 主函数(Main function)
def main():
    global image   # 声明全局变量 image(Declaring global variable ‘image’. )
    cam_num = 0   # 设置摄像头编号(Set the camera number)
    cap = blinx_open_camera(cam_num)   # 打开摄像头(Open camera)
    try:
        while True:
            ret, frame = cap.read()   # 读取摄像头的一帧图像(Set the camera
number)
            if not ret:   # 如果读取帧未成功(If the frame is not read
successfully)
                print("获取帧失败(Failed to fetch frame)")
                break
            cv2.imshow("Camera", frame)   # 显示图像(Display image)
            image = frame   # 保存当前帧以便进行检测(The current frame is
saved for detection)
            key = cv2.waitKey(30) & 0xFF   # 等待按键输入，每隔 30ms 检
查一次(Wait for the key to be entered, and check every 30ms)
            if key == ord('d'):   # 如果按下'd'键(Press the 'D' key)
                blinx_start_detection(image)   # 开始进行物体水果识别
(Perform object fruit recognition)
            elif key == ord('q'):   # 如果按下'q'键(Press the 'Q' key)
```

```
                break  # 退出循环(Exit the loop)
        finally:
            cap.release()  # 释放摄像头(release the camera)
            cv2.destroyAllWindows()  # 关闭所有 OpenCV 窗口(Close all
OpenCV windows)
```

5.10 Running the main program

If the script is run as the main program, the "blinx_send_data" thread and the main program are started. The "blinx_send_data" thread is responsible for processing the object recognition results and controlling the robotic arm, and the main program is responsible for reading the camera image and triggering the recognition.

The example code is as follows:

Proceed with the program file "基于视觉的机械臂水果分类.ipynb"

```
# 程序入口(Program Entrance)
if __name__ == "__main__":
    # 启动发送数据线程(Start the send data thread)
    thread_send = threading.Thread(target=blinx_send_data)
    thread_send.start()
    main()  # 运行主程序(Run the main program)
```

5.11 Program running effect

Note that the cube should be placed squarely towards the camera, not diagonally.