Kibo-RPC 範例程式碼解析

程式概述

這是第六屆 Kibo Robot Programming Challenge (Kibo-RPC) 的範例程式碼，用於控制在國際太空站上的 Astrobee 機器人執行尋寶任務。

完整程式碼

```
package jp.jaxa.iss.kibo.rpc.sampleapk;

import jp.jaxa.iss.kibo.rpc.api.KiboRpcService;

import gov.nasa.arc.astrobee.types.Point;
import gov.nasa.arc.astrobee.types.Quaternion;

import org.opencv.core.Mat;

/**
 * Class meant to handle commands from the Ground Data System and execute them in
Astrobee.
 */

public class YourService extends KiboRpcService {
    @Override
    protected void runPlan1(){
        // The mission starts.
        api.startMission();
        // Move to a point.
        Point point = new Point(10.9d, -9.92284d, 5.195d);
        Quaternion quaternion = new Quaternion(0f, 0f, -0.707f, 0.707f);
        api.moveTo(point, quaternion, false);

        // Get a camera image.
        Mat image = api.getMatNavCam();

        /*
**********************************************************************
*/
        /* Write your code to recognize the type and number of landmark items in
each area! */
        /* If there is a treasure item, remember it.
*/
        /*
**********************************************************************
*/

        // When you recognize landmark items, let's set the type and number.
        api.setAreaInfo(1, "item_name", 1);
```

```
        /* ************************************************** */
        /* Let's move to each area and recognize the items. */
        /* ************************************************** */

        // When you move to the front of the astronaut, report the rounding
completion.
        point = new Point(11.143d, -6.7607d, 4.9654d);
        quaternion = new Quaternion(0f, 0f, 0.707f, 0.707f);
        api.moveTo(point, quaternion, false);
        api.reportRoundingCompletion();

        /* ********************************************************** */
        /* Write your code to recognize which target item the astronaut has. */
        /* ********************************************************** */

        // Let's notify the astronaut when you recognize it.
        api.notifyRecognitionItem();

        /*
**********************************************************************************
********************* */
        /* Write your code to move Astrobee to the location of the target item
(what the astronaut is looking for) */
        /*
**********************************************************************************
********************* */

        // Take a snapshot of the target item.
        api.takeTargetItemSnapshot();
    }

    @Override
    protected void runPlan2(){
       // write your plan 2 here.
    }

    @Override
    protected void runPlan3(){
        // write your plan 3 here.
    }

    // You can add your method.
    private String yourMethod(){
        return "your method";
    }
}
```

程式結構解析

1. 套件宣告與匯入

```
package jp.jaxa.iss.kibo.rpc.sampleapk;

import jp.jaxa.iss.kibo.rpc.api.KiboRpcService;
import gov.nasa.arc.astrobee.types.Point;
import gov.nasa.arc.astrobee.types.Quaternion;
import org.opencv.core.Mat;
```

說明：

- `package jp.jaxa.iss.kibo.rpc.sampleapk` - 定義程式所屬的套件
- `KiboRpcService` - Kibo-RPC 的核心服務類別，所有參賽程式都必須繼承此類別
- `Point` - 表示三維空間座標的類別
- `Quaternion` - 表示三維空間旋轉的四元數類別
- `Mat` - OpenCV 的影像矩陣類別，用於影像處理

2. 類別定義

```
public class YourService extends KiboRpcService {
```

說明：

- 繼承 `KiboRpcService` 類別，獲得控制 Astrobee 機器人的能力
- 類別名稱 `YourService` 可以自訂，但必須繼承 `KiboRpcService`

3. 主要執行計畫 (runPlan1)

3.1 任務開始

```
api.startMission();
```

說明：

- 從對接站解除對接，正式開始任務計時
- 這是每個任務的必要第一步

3.2 移動到指定位置

```
Point point = new Point(10.9d, -9.92284d, 5.195d);
Quaternion quaternion = new Quaternion(0f, 0f, -0.707f, 0.707f);
api.moveTo(point, quaternion, false);
```

說明：

- `Point(10.9d, -9.92284d, 5.195d)` - 創建三維座標點 (x=10.9, y=-9.92284, z=5.195)

- `Quaternion(0f, 0f, -0.707f, 0.707f)` - 創建表示方向的四元數
- `api.moveTo()` - 移動機器人到指定位置和方向
- 第三個參數 `false` 表示不在日誌中列印機器人位置

3.3 獲取相機影像

```
Mat image = api.getMatNavCam();
```

說明：

- 從導航相機獲取影像，返回 OpenCV 的 Mat 格式
- 影像尺寸為 1280 x 960 像素，格式為 CV8UC1（灰階）

3.4 影像識別區塊（需要實作）

```
/*
********************************************************************************
*/
/* Write your code to recognize the type and number of landmark items in each
area! */
/* If there is a treasure item, remember it.
*/
/*
********************************************************************************
*/
```

任務說明：

- 需要識別各區域中的地標物品類型和數量
- 地標物品包括：coin, compass, coral, fossil, key, letter, shell, treasure_box
- 如果發現寶藏物品，需要記住位置

3.5 設定區域資訊

```
api.setAreaInfo(1, "item_name", 1);
```

說明：

- 第一個參數：區域編號 (1-4)
- 第二個參數：物品名稱
- 第三個參數：物品數量
- 此處的 "item_name" 需要替換為實際識別到的物品名稱

3.6 移動到太空人位置

```
point = new Point(11.143d, -6.7607d, 4.9654d);
quaternion = new Quaternion(0f, 0f, 0.707f, 0.707f);
api.moveTo(point, quaternion, false);
api.reportRoundingCompletion();
```

說明：

- 移動到太空人面前的指定座標
- `reportRoundingCompletion()` - 報告巡邏完成，必須在正確位置執行

3.7 識別目標物品（需要實作）

```
/* ********************************************************** */
/* Write your code to recognize which target item the astronaut has. */
/* ********************************************************** */
```

任務說明：

- 需要從太空人手中的線索影像識別真正的目標寶藏

3.8 通知識別完成

```
api.notifyRecognitionItem();
```

說明：

- 告知太空人已經識別出目標物品

3.9 移動到目標位置（需要實作）

```
/*
**********************************************************************************
********************* */
/* Write your code to move Astrobee to the location of the target item (what the
astronaut is looking for) */
/*
**********************************************************************************
********************* */
```

任務說明：

- 根據識別結果，移動到真正寶藏的位置

3.10 拍攝目標物品

```
api.takeTargetItemSnapshot();
```

說明：

- 拍攝目標物品的快照，完成任務
- 需要在正確的角度（30度視角內）和距離（0.9公尺內）執行

4. 備用計畫

```
@Override
protected void runPlan2(){
   // write your plan 2 here.
}

@Override
protected void runPlan3(){
    // write your plan 3 here.
}
```

說明：

- `runPlan2` 和 `runPlan3` 是備用計畫
- 在網路模擬器中只會執行 `runPlan1`
- 在本地機器上可以選擇執行哪個計畫

5. 自訂方法

```
private String yourMethod(){
    return "your method";
}
```

說明：

- 可以添加自訂的私有方法來組織程式碼
- 建議將影像處理、路徑規劃等功能分離到不同方法中

程式執行流程

1. **初始化** - 繼承 KiboRpcService，獲得 API 控制能力
2. **開始任務** - 從對接站解除對接
3. **移動並拍攝** - 移動到指定位置，獲取相機影像
4. **影像識別** - 識別區域中的物品（需要實作）
5. **設定資訊** - 記錄識別結果
6. **報告完成** - 移動到太空人處，報告巡邏完成
7. **識別目標** - 識別太空人的目標物品（需要實作）

8. **移動取寶** - 移動到目標位置（需要實作）
9. **完成任務** - 拍攝目標物品快照