Kibo-RPC 完整四區域巡邏與物品識別教學

新增功能概述

這個版本是前面所有功能的整合，實現了完整的四區域自動巡邏系統。程式能夠依序訪問所有區域，在每個區域進行物品識別，並且加入了記憶體管理和資源釋放機制，使程式更加穩定和實用。

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;

// new imports
import android.util.Log;

import java.util.List;
import java.util.ArrayList;


// new OpenCV imports
import org.opencv.aruco.Dictionary;
import org.opencv.aruco.Aruco;
import org.opencv.calib3d.Calib3d;
import org.opencv.core.CvType;
import java.io.InputStream;
import java.io.IOException;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import org.opencv.android.Utils;
import org.opencv.imgproc.Imgproc;
import org.opencv.core.Size;
import org.opencv.core.Core;


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

public class YourService extends KiboRpcService {

    // The TAG is used for logging.
    // You can use it to check the log in the Android Studio.
    private final String TAG = this.getClass().getSimpleName();
```

```
    // Type all the image in the template folder.
    //'coin.png', 'compass.png', 'coral.png', 'crystal.png',
    // 'diamond.png', 'emerald.png', 'fossil.png', 'key.png',
    // 'letter.png', 'shell.png', 'treasure_box.png']
    private final String[] TEMPLATE_FILE_NAMES = {
            "coin.png",
            "compass.png",
            "coral.png",
            // "crystal.png",
            // "diamond.png",
            // "emerald.png",
            "fossil.png",
            "key.png",
            "letter.png",
            "shell.png",
            "treasure_box.png"
    };

    private final String[] TEMPLATE_NAMES = {
            "coin",
            "compass",
            "coral",
            // "crystal",
            // "diamond",
            // "emerald",
            "fossil",
            "key",
            "letter",
            "shell",
            "treasure_box"
    };

    private final Point[] AREA_POINTS = {
        new Point(10.9d, -10.0000d, 5.195d),    // Area 1
        new Point(10.925d, -8.875d, 4.602d),    // Area 2
        new Point(10.925d, -7.925d, 4.60093d),    // Area 3
        new Point(10.766d, -6.852d, 4.945d)     // Area 4
    };

    private final Quaternion[] AREA_QUATERNIONS = {
        new Quaternion(0f, 0f, -0.707f, 0.707f),    // Area 1
        new Quaternion(0f, 0.707f, 0f, 0.707f),     // Area 2
        new Quaternion(0f, 0.707f, 0f, 0.707f),     // Area 3
        new Quaternion(0f, 0f, 1f, 0f)      // Area 4
    };

    @Override
    protected void runPlan1(){
        // Log the start of the mission.
        Log.i(TAG, "Start mission");
        // The mission starts.
        api.startMission();
```

```
        // Prepare the variables to store treasure item locations
        String[] foundItems = new String[AREA_POINTS.length];
        int[] itemCounts = new int[AREA_POINTS.length];

        // Load template images
        Mat[] templates = new Mat[TEMPLATE_FILE_NAMES.length];
        for (int i = 0; i < TEMPLATE_FILE_NAMES.length; i++) {
            try{
                // Open the template image file in Bitmap from the file name and
convert to Mat
                InputStream inputStream =
getAssets().open(TEMPLATE_FILE_NAMES[i]);
                Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
                Mat mat = new Mat();
                Utils.bitmapToMat(bitmap, mat);

                // Convert the Bitmap to grayscale
                Imgproc.cvtColor(mat, mat, Imgproc.COLOR_BGR2GRAY);

                // Assign to an array of templates
                templates[i] = mat;

                // Release the InputStream
                inputStream.close();

            } catch (IOException e) {
                Log.e(TAG, "Error loading template image: " +
TEMPLATE_FILE_NAMES[i], e);
                e.printStackTrace();
            }
        }


        // Visit each area
        for (int areaId = 0; areaId < AREA_POINTS.length; areaId++) {
            // Move to the current area
            api.moveTo(AREA_POINTS[areaId], AREA_QUATERNIONS[areaId], false);

            // Get a camera image
            Mat image = api.getMatNavCam();
            // Save the image for debugging
            api.saveMatImage(image, "area_" + (areaId + 1) + ".png");

            /* ********************************************** */
            /* Start image recognition process */
            /* ********************************************** */
            // AR tag detection
            Dictionary dictionary =
Aruco.getPredefinedDictionary(Aruco.DICT_5X5_250);
            List<Mat> corners = new ArrayList<>();
            Mat ids = new Mat();
            Aruco.detectMarkers(image, dictionary, corners, ids);

            // Undistort the image
```

```
            Mat cameraMatrix = new Mat(3,3,CvType.CV_64F);
            cameraMatrix.put(0, 0, api.getNavCamIntrinsics()[0]);

            Mat cameraCoefficients = new Mat(1, 5, CvType.CV_64F);
            cameraCoefficients.put(0, 0, api.getNavCamIntrinsics()[1]);
            cameraCoefficients.convertTo(cameraCoefficients, CvType.CV_64F);

            Mat undistortImg = new Mat();
            Calib3d.undistort(image, undistortImg, cameraMatrix,
cameraCoefficients);

            // Number of matches for each template
            int[] numMatches = new int[TEMPLATE_FILE_NAMES.length];

            // Get the number of template matches
            for (int tempNum = 0; tempNum < templates.length; tempNum++) {
                // number of matches
                int matchCount = 0;
                // Coordinates of the match location
                List<org.opencv.core.Point> matchLocations = new ArrayList<>();

                // Load the template image
                Mat template = templates[tempNum].clone();
                // Target image is the undistorted image
                Mat targetImg = undistortImg.clone();

                // Pattern matching
                int widthMin = 20; //[px]
                int widthMax = 100; //[px]
                int changeWidth = 5; //[px]
                int changeAngle = 45; //[px]

                for (int size=widthMin; size <= widthMax; size += changeWidth){
                    for (int angle=0; angle<360; angle+=changeAngle){
                        // Resize the template image
                        Mat resizedTemplate = scalingresizeImage(template, size);
                        // Rotate the template image
                        Mat rotatedTemplate = rotImg(resizedTemplate, angle);

                        // Perform template matching
                        Mat result = new Mat();
                        Imgproc.matchTemplate(targetImg, rotatedTemplate, result,
Imgproc.TM_CCOEFF_NORMED);

                        // Thresholding
                        double threshold = 0.7;
                        Core.MinMaxLocResult mmlr = Core.minMaxLoc(result);

                        double maxVal = mmlr.maxVal;

                        if (maxVal >= threshold) {
                            // Create a mask for the detected region
                            Mat thresholdResult = new Mat();
                            Imgproc.threshold(result, thresholdResult, threshold,
```

```
1, Imgproc.THRESH_TOZERO);

                            // Get coordinates of the detected region
                            for(int y=0; y<thresholdResult.rows(); y++){
                                for(int x=0; x<thresholdResult.cols(); x++){
                                    if(thresholdResult.get(y, x)[0] > 0){
                                        matchLocations.add(new
org.opencv.core.Point(x, y));
                                    }
                                }
                            }

                            // Release resources
                            thresholdResult.release();
                        }

                        // Release resources
                        result.release();
                        rotatedTemplate.release();
                        resizedTemplate.release();
                    }
                }

                List<org.opencv.core.Point> filteredMatches =
removeDuplicates(matchLocations);
                matchCount += filteredMatches.size();

                // Number of matches
                numMatches[tempNum] = matchCount;

                // Release resources
                template.release();
                targetImg.release();
            }

            // Find the most matched template
            int mostMatchTemplateNum = getMxIndex(numMatches);

            // Store the results
            foundItems[areaId] = TEMPLATE_NAMES[mostMatchTemplateNum];
            itemCounts[areaId] = numMatches[mostMatchTemplateNum];
            /* ****************************************** */
            /* End image recognition process  and Reprot*/
            // Report what was found in this area
            api.setAreaInfo(areaId + 1, foundItems[areaId], itemCounts[areaId]);

            // Log the results for debugging
            Log.i(TAG, "Area " + (areaId + 1) + ": Found " + itemCounts[areaId] +
" of " + foundItems[areaId]);

            // Release resources
            image.release();
            undistortImg.release();
            cameraMatrix.release();
```

```
            cameraCoefficients.release();
            ids.release();
            for (Mat corner : corners) {
                corner.release();
            }

        }

        // Move to the astronaut for reporting
        Point astronautPoint = new Point(11.143, -6.7607, 4.9654);
        Quaternion astronautQuaternion = new Quaternion(0f, 0f, 0.707f, 0.707f);
        api.moveTo(astronautPoint, astronautQuaternion, false);
        api.reportRoundingCompletion();

        // Get information about the target item from the astronaut
        // Mat targetImage = api.getMatNavCam();
        // api.saveMatImage(targetImage, "target_item.png");

        // TODO: Implement logic to identify the target item from astronaut's
information
        // For now, let's assume the target is in the first area
        // String targetItem = TEMPLATE_NAMES[0]; // Replace with actual target
item identification

        // Notify that we recognized the target item
        api.notifyRecognitionItem();

        // // Find which area contains the target item and move there
        // int targetArea = 0; // Default to first area if not found
        // for (int i = 0; i < foundItems.length; i++) {
        //     if (foundItems[i].equals(targetItem)) {
        //         targetArea = i;
        //         break;
        //     }
        // }

        // Move to the area with the target item
        // api.moveTo(AREA_POINTS[targetArea], AREA_QUATERNIONS[targetArea],
false);

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

        // Release resources
        // targetImage.release();
        // for (Mat template : templates) {
        //     if (template != null) {
        //         template.release();
        //     }
        // }
    }

    @Override
    protected void runPlan2(){
```

```
        // write your plan 2 here.
    }

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

    private Mat scalingresizeImage(Mat image, int width) {
        int height = (int) ((double) image.rows() / image.cols() * width);
        // Create a new Mat object to hold the resized image
        Mat resizedImage = new Mat();

        // Resize the image using the specified width and height
        Imgproc.resize(image, resizedImage, new Size(width, height));

        return resizedImage;
    }

    private Mat rotImg(Mat img, int angle) {
        // Get the center of the image
        org.opencv.core.Point center = new org.opencv.core.Point(img.cols() / 2,
img.rows() / 2);

        // Create a rotation matrix
        Mat rotMat = Imgproc.getRotationMatrix2D(center, angle, 1.0);

        // Create a new Mat object to hold the rotated image
        Mat rotatedImg = new Mat();

        // Rotate the image using the rotation matrix
        Imgproc.warpAffine(img, rotatedImg, rotMat, img.size());

        // Release resources
        rotMat.release();

        return rotatedImg;
    }

    private List<org.opencv.core.Point>
removeDuplicates(List<org.opencv.core.Point> points) {
        double length = 10; // within 10 px
        List<org.opencv.core.Point> uniquePoints = new ArrayList<>();
        for (org.opencv.core.Point point : points) {
            boolean isIncluded = false;
            for (org.opencv.core.Point uniquePoint : uniquePoints) {
                double distance = calculateDistance(point, uniquePoint);

                if (distance <= length){
                    isIncluded = true;
                    break;
                }
            }
```

```
            if (!isIncluded) {
                uniquePoints.add(point);
            }
        }
        return uniquePoints;
    }

    private double calculateDistance(org.opencv.core.Point p1,
org.opencv.core.Point p2) {
        // Calculate the distance between two points using the Euclidean distance
formula
        return Math.sqrt(Math.pow(p2.x - p1.x, 2) + Math.pow(p2.y - p1.y, 2));
    }

    private int getMxIndex(int[] array){
        int max = 0;
        int maxIndex = 0;

        for (int i = 0; i < array.length; i++) {
            if (array[i] > max) {
                max = array[i];
                maxIndex = i;
            }
        }
        return maxIndex;
    }
}
```

新增的程式碼解析

1. 區域座標與方向定義

```
private final Point[] AREA_POINTS = {
    new Point(10.9d, -10.0000d, 5.195d),    // Area 1
    new Point(10.925d, -8.875d, 4.602d),    // Area 2
    new Point(10.925d, -7.925d, 4.60093d),   // Area 3
    new Point(10.766d, -6.852d, 4.945d)     // Area 4
};

private final Quaternion[] AREA_QUATERNIONS = {
    new Quaternion(0f, 0f, -0.707f, 0.707f),    // Area 1
    new Quaternion(0f, 0.707f, 0f, 0.707f),     // Area 2
    new Quaternion(0f, 0.707f, 0f, 0.707f),     // Area 3
    new Quaternion(0f, 0f, 1f, 0f)              // Area 4
};
```

程式碼解析：

- `AREA_POINTS` - 定義四個區域的精確座標位置

- `AREA_QUATERNIONS` - 定義每個區域對應的相機朝向
- 陣列索引對應區域編號（0 = Area 1, 1 = Area 2, 等等）

座標系統說明：

- 座標基於 ISS 的固定坐標系統
- 每個區域都有最佳的觀察位置和角度
- 方向設定確保相機能正確對準目標區域

2. 結果儲存變數初始化

```
// Prepare the variables to store treasure item locations
String[] foundItems = new String[AREA_POINTS.length];
int[] itemCounts = new int[AREA_POINTS.length];
```

程式碼解析：

- `foundItems` - 儲存每個區域識別出的物品名稱
- `itemCounts` - 儲存每個區域對應物品的數量
- 陣列大小與區域數量相同，便於後續處理

3. 核心巡邏迴圈

```
// Visit each area
for (int areaId = 0; areaId < AREA_POINTS.length; areaId++) {
    // Move to the current area
    api.moveTo(AREA_POINTS[areaId], AREA_QUATERNIONS[areaId], false);

    // Get a camera image
    Mat image = api.getMatNavCam();
    // Save the image for debugging
    api.saveMatImage(image, "area_" + (areaId + 1) + ".png");
```

程式碼解析：

- `for` 迴圈依序訪問四個區域
- `areaId` 從 0 到 3，對應 Area 1 到 Area 4
- 每個區域都會拍攝影像並以區域編號命名儲存

檔案命名規則：

- area_1.png, area_2.png, area_3.png, area_4.png
- 便於後續分析每個區域的影像品質

4. 改進的記憶體管理

```

```

```
// Release resources
thresholdResult.release();

// Release resources
result.release();
rotatedTemplate.release();
resizedTemplate.release();
```

程式碼解析：

- 在每次處理完成後立即釋放 Mat 物件
- 避免記憶體洩漏和資源耗盡
- 提高程式執行的穩定性

記憶體管理最佳實務：

```
// 在迴圈內及時釋放臨時變數
// 在方法結束前釋放主要變數
// 避免累積大量未釋放的 Mat 物件
```

5. 結果儲存與回報

```
// Find the most matched template
int mostMatchTemplateNum = getMxIndex(numMatches);

// Store the results
foundItems[areaId] = TEMPLATE_NAMES[mostMatchTemplateNum];
itemCounts[areaId] = numMatches[mostMatchTemplateNum];

// Report what was found in this area
api.setAreaInfo(areaId + 1, foundItems[areaId], itemCounts[areaId]);

// Log the results for debugging
Log.i(TAG, "Area " + (areaId + 1) + ": Found " + itemCounts[areaId] + " of " +
foundItems[areaId]);
```

程式碼解析：

- 即時儲存每個區域的識別結果
- 使用 `areaId + 1` 將陣列索引轉換為實際區域編號
- 同時記錄日誌便於除錯和監控

6. 區域間資源清理

```
// Release resources
image.release();
```

```
undistortImg.release();
cameraMatrix.release();
cameraCoefficients.release();
ids.release();
for (Mat corner : corners) {
    corner.release();
}
```

程式碼解析：

- 在每個區域處理完成後釋放所有相關資源
- 特別注意釋放集合中的每個 Mat 物件
- 確保下一個區域有足夠的記憶體空間

7. 任務完成流程

```
// Move to the astronaut for reporting
Point astronautPoint = new Point(11.143, -6.7607, 4.9654);
Quaternion astronautQuaternion = new Quaternion(0f, 0f, 0.707f, 0.707f);
api.moveTo(astronautPoint, astronautQuaternion, false);
api.reportRoundingCompletion();

// TODO: Implement logic to identify the target item from astronaut's information
// For now, let's assume the target is in the first area
// String targetItem = TEMPLATE_NAMES[0]; // Replace with actual target item
identification

// Notify that we recognized the target item
api.notifyRecognitionItem();

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

程式碼解析：

- 所有區域巡邏完成後移動到太空人位置
- 包含 TODO 註解標示待實現的功能
- 暫時跳過目標物品識別，直接完成任務

改進的輔助方法

1. 增強的圖像旋轉方法

```
private Mat rotImg(Mat img, int angle) {
    // Get the center of the image
    org.opencv.core.Point center = new org.opencv.core.Point(img.cols() / 2,
img.rows() / 2);

    // Create a rotation matrix
```

```
    Mat rotMat = Imgproc.getRotationMatrix2D(center, angle, 1.0);

    // Create a new Mat object to hold the rotated image
    Mat rotatedImg = new Mat();

    // Rotate the image using the rotation matrix
    Imgproc.warpAffine(img, rotatedImg, rotMat, img.size());

    // Release resources
    rotMat.release();

    return rotatedImg;
}
```

改進點：

- 新增 `rotMat.release()` 釋放旋轉矩陣
- 避免在大量旋轉操作中累積記憶體使用

實際應用建議

1. 完整的錯誤處理機制

```
private boolean processArea(int areaId) {
    try {
        // Move to area
        Result moveResult = api.moveTo(AREA_POINTS[areaId],
AREA_QUATERNIONS[areaId], false);
        if (!moveResult.hasSucceeded()) {
            Log.e(TAG, "Failed to move to area " + (areaId + 1));
            return false;
        }

        // Get image
        Mat image = api.getMatNavCam();
        if (image == null || image.empty()) {
            Log.e(TAG, "Failed to capture image in area " + (areaId + 1));
            return false;
        }

        // Process image...
        return true;

    } catch (Exception e) {
        Log.e(TAG, "Error processing area " + (areaId + 1), e);
        return false;
    }
}
```

2. 動態參數調整

```
private class AreaSpecificParams {
    public int widthMin, widthMax, changeWidth, changeAngle;
    public double threshold;

    public AreaSpecificParams(int area) {
        switch (area) {
            case 0: // Area 1 - 較遠距離
                widthMin = 15; widthMax = 60; threshold = 0.65;
                break;
            case 1: // Area 2 - 中距離
                widthMin = 20; widthMax = 80; threshold = 0.7;
                break;
            case 2: // Area 3 - 中距離
                widthMin = 20; widthMax = 80; threshold = 0.7;
                break;
            case 3: // Area 4 - 較近距離
                widthMin = 30; widthMax = 100; threshold = 0.75;
                break;
        }
        changeWidth = 3;
        changeAngle = 30;
    }
}
```

3. 結果驗證機制

```
private boolean validateResults(String[] foundItems, int[] itemCounts) {
    for (int i = 0; i < foundItems.length; i++) {
        if (foundItems[i] == null || itemCounts[i] < 0) {
            Log.w(TAG, "Invalid result in area " + (i + 1));
            return false;
        }

        if (itemCounts[i] == 0) {
            Log.w(TAG, "No items found in area " + (i + 1));
        }
    }
    return true;
}
```

性能優化建議

1. 模板預處理

```
private void preprocessTemplates(Mat[] templates) {
    for (int i = 0; i < templates.length; i++) {
        if (templates[i] != null) {
```

```
                // 正規化模板亮度
                Mat normalized = new Mat();
                Core.normalize(templates[i], normalized, 0, 255, Core.NORM_MINMAX);
                templates[i].release();
                templates[i] = normalized;
            }
        }
    }
```

2. 平行處理考量

```
// 注意：在 Astrobee 上不建議使用多線程
// 但可以考慮批次處理來優化性能
private int[] batchProcessTemplates(Mat targetImg, Mat[] templates, int startIdx,
int endIdx) {
    int[] results = new int[endIdx - startIdx];

    for (int i = startIdx; i < endIdx; i++) {
        results[i - startIdx] = processTemplate(targetImg, templates[i]);
    }

    return results;
}
```