Kibo-RPC 圖像模板匹配與物品識別教學

新增功能概述

這個版本的程式碼在前面 AR 標籤檢測和影像矯正的基礎上，新增了完整的圖像模板匹配系統，能夠自動識別各種物品類型並計算數量。這是 Kibo-RPC 任務中最核心的功能，透過模板匹配技術來識別太空站中的地標物品和寶藏物品。

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"
    };




    @Override
    protected void runPlan1(){

        // Log the start of the mission.
        Log.i(TAG, "Start mission");
        // 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();
```

```
        // Save the image to a file.
        api.saveMatImage(image, "test.png");


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

        //
        /**
         * Retrieves a predefined Aruco dictionary for 6x6 markers containing 250
distinct patterns.
         * This dictionary is used for detecting and tracking Aruco markers in
images.
         *
         * The call to Aruco.getPredefinedDictionary(Aruco.DICT_6X6_250) selects a
standard set of marker patterns,
         * making it easier to consistently identify markers during image
processing.
         */
        Dictionary dictionary = Aruco.getPredefinedDictionary(Aruco.DICT_5X5_250);

        // Detect markers in the image using the specified dictionary.
        // The detectMarkers function analyzes the image and identifies the
locations of Aruco markers.
        // The detected markers are stored in the corners list.
        // The corners list contains the coordinates of the detected markers in
the image.

        List<Mat> corners = new ArrayList<>();
        Mat ids = new Mat();
        // The ids list contains the IDs of the detected markers.
        Aruco.detectMarkers(image, dictionary, corners, ids);



        // Undistort the detected markers using the camera matrix and distortion
coefficients.
        // getNavCamIntrinsics will  return  ( cameraMatrix,
distortionCoefficients )

        // Get camera matrix (linear)
        Mat cameraMatrix = new Mat(3,3,CvType.CV_64F);
        cameraMatrix.put(0, 0, api.getNavCamIntrinsics()[0]);


        // Get len distortion parameters (non linear)
```

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

        // Undistort the detected markers using the camera matrix and distortion
coefficients.
        Mat undistortImg = new Mat();
        Calib3d.undistort(image, undistortImg, cameraMatrix, cameraCoefficients);


        /*
*********************************************************************************
                                        Pattern Matching

*********************************************************************************
*/


        // 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
                // The BitmapFactory.decodeStream method is used to decode the
image file into a Bitmap object.
                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();
            }
        }

        // 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 loaction
            List<org.opencv.core.Point> matchLocations = new ArrayList<>();


            // Load the template image
            Mat template = templates[tempNum].clone();
            // Convert the template image to grayscale
            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){
                                    //matchCount++;
                                    matchLocations.add(new
org.opencv.core.Point(x, y));
                                }
                            }
                        }

                    }
                }
```

```
            }

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

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

        // Start Pattern Matching
        // Load template images from the template folder


        // When you recognize landmark items, let’s set the type and number.
        int mostMatchTemplateNum = getMxIndex(numMatches);
        api.setAreaInfo(1, TEMPLATE_NAMES[mostMatchTemplateNum],
numMatches[mostMatchTemplateNum]);

        // put

        /* ************************************************** */
        /* 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";
    }

    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());

        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. 新增匯入

```
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;
```

說明：

- `InputStream` - 用於讀取資產檔案的輸入串流

- `IOException` - 處理檔案讀取例外
- `Bitmap`, `BitmapFactory` - Android 圖像處理類別
- `Utils` - OpenCV 與 Android 圖像格式轉換工具
- `Imgproc` - OpenCV 影像處理模組
- `Size` - OpenCV 尺寸類別
- `Core` - OpenCV 核心功能模組

2. 模板檔案定義

```
// 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"
};
```

程式碼解析：

- `TEMPLATE_FILE_NAMES` - 儲存模板圖像檔案名稱的陣列
- `TEMPLATE_NAMES` - 對應的物品名稱陣列，用於 API 回報
- **注意：**寶藏物品（crystal, diamond, emerald）被註解掉，只檢測地標物品

設計考量：

- 兩個陣列必須保持相同順序和大小

- 檔案名稱包含副檔名，物品名稱不包含
- 可以透過註解控制要檢測的物品類型

3. 模板圖像載入

```
// 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
        // The BitmapFactory.decodeStream method is used to decode the image file
into a Bitmap object.
        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();
    }
}
```

程式碼解析：

- `getAssets().open()` - 從 assets 資料夾載入圖像檔案
- `BitmapFactory.decodeStream()` - 將檔案串流解碼為 Bitmap
- `Utils.bitmapToMat()` - 轉換 Android Bitmap 為 OpenCV Mat
- `Imgproc.cvtColor()` - 轉換為灰階圖像以提高匹配效率

錯誤處理：

- 使用 try-catch 處理檔案讀取例外
- 記錄詳細的錯誤資訊用於除錯

4. 多尺度多角度模板匹配

```
// 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 loaction
    List<org.opencv.core.Point> matchLocations = new ArrayList<>();

    // Load the template image
    Mat template = templates[tempNum].clone();
    // Convert the template image to grayscale
    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){
                            //matchCount++;
                            matchLocations.add(new org.opencv.core.Point(x, y));
                        }
                    }
                }
            }
        }
    }
```

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

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

程式碼解析：

- **多尺度搜尋：**寬度從 20 到 100 像素，每次增加 5 像素
- **多角度搜尋：**從 0 到 360 度，每次旋轉 45 度
- **模板匹配：**使用正規化相關係數匹配方法
- **閾值過濾：**只接受相似度 0.7 以上的匹配結果

匹配參數說明：

```
int widthMin = 20;       // 最小搜尋尺寸
int widthMax = 100;      // 最大搜尋尺寸
int changeWidth = 5;     // 尺寸變化步長
int changeAngle = 45;    // 角度變化步長
double threshold = 0.7; // 匹配閾值
```

5. 結果識別和回報

```
// When you recognize landmark items, let's set the type and number.
int mostMatchTemplateNum = getMxIndex(numMatches);
api.setAreaInfo(1, TEMPLATE_NAMES[mostMatchTemplateNum],
numMatches[mostMatchTemplateNum]);
```

程式碼解析：

- `getMxIndex()` - 找出匹配數量最多的模板索引
- `api.setAreaInfo()` - 回報識別結果給系統

新增的輔助方法解析

1. 圖像縮放方法

```
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;
}
```

功能說明：

- 根據指定寬度等比例縮放圖像
- 保持原始長寬比例
- 使用 OpenCV 的 resize 函式

2. 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());

    return rotatedImg;
}
```

功能說明：

- 以圖像中心為軸心旋轉
- 使用仿射變換實現旋轉
- 保持原始圖像尺寸

3. 重複點移除方法

```
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;
    }
```

功能說明：

- 移除距離 10 像素內的重複檢測點
- 避免同一物品被重複計算
- 提高檢測精確度

4. 距離計算方法

```
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));
}
```

功能說明：

- 使用歐幾里得距離公式
- 計算兩點間的直線距離

5. 最大值索引查找方法

```
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;
}
```

功能說明：

- 找出陣列中最大值的索引
- 用於確定匹配數量最多的模板

演算法流程總結

1. **預處理階段**
    - 載入所有模板圖像
    - 轉換為灰階格式
    - 影像去畸變處理
2. **匹配階段**
    - 對每個模板進行多尺度多角度搜尋
    - 計算正規化相關係數
    - 應用閾值過濾
3. **後處理階段**
    - 移除重複檢測點
    - 統計每種物品的數量
    - 選擇匹配數量最多的物品類型
4. **結果回報**
    - 回報識別的物品類型和數量

這個模板匹配系統提供了強健的物品識別能力，能夠應對不同尺寸、角度和光照條件下的物品檢測需求，為 Kibo-RPC 任務的成功執行提供了關鍵技術支援。