**Bliffoscope**

Application which scans for target images against Test Data.

**Prerequisites**

1. Java Runtime (JDK for Development)
2. Gradle Build Tool (<https://gradle.org/install/#with-a-package-manager>)
   1. Extract the Gradle zip file.
   2. Set the environment path variable pointing to ‘**bin’** folder of Gradle.

**How to run application**

1. Go to project folder and run below command.

gradle bootRun

**How to make build**

1. Run gradle build command in project folder.

gradle build

**Notes**

1. **On start** of application, test data(TestData.blf) is analyzed against slime torpedo(SlimeTorpedo.blf) and Starship (Starship.blf), with threshold 70 as default and Result is presented in **console**. (file are present in resource folder)
2. **Threshold** : Minimum percentage match required to consider a subset of test data as target.
3. **Coordinates:** target matrix first cell position.
4. Web interface to test the application is available at <http://127.0.0.1:8090/>

**Implementation**

1. **findTargets** method finds all targets in given test data image(Boolean matrix), It basically computes score for each sub matrix of with height and length that of targetImage.

**public** List<Target> findTargets(Image sourceImage, Image targetImage, **int** threshold) {  
 List<Target> targets;  
  
 **if** (sourceImage == **null**) {  
 **throw new** NotFoundException(100, **"Source image not found"**);  
 } **else if** (targetImage == **null**) {  
 **throw new** NotFoundException(101, **"Target image not found"**);  
 } **else if** (threshold < 0 || threshold > 100) {  
 **throw new** InvalidDataException(102, **"Threshold should be between 0 and 100"**);  
 } **else** {  
 targets = **new** ArrayList<>();  
 **float** targetArea = targetImage.getHeight() \* targetImage.getWidth();  
 **for** (**int** row = 0; row <= sourceImage.getHeight() - targetImage.getHeight(); row++) {  
 **for** (**int** col = 0; col <= sourceImage.getWidth() - targetImage.getWidth(); col++) {  
 **int** score = *getScoreForSubSet*(sourceImage, targetImage, row, col);  
 **float** percentageMatch = ((score / targetArea) \* 100);  
 **if** (percentageMatch >= threshold) {  
 Target target = **new** Target();  
 target.setType(targetImage.getName());  
 target.setCoordinates(**new** Coordinates(row, col));  
 target.setPercentageMatch(percentageMatch);  
 targets.add(target);  
 }  
 }  
 }  
 }  
  
 **return** targets;  
}

1. **getScoreForSubSet** method for give sub matrix and position (row, col) of test data, it computes score, by comparing each pixel against targetImage.

**private static int** getScoreForSubSet(Image sourceImage, Image targetImage, **int** row, **int** col) {  
 **int** score = 0;  
 **for** (**int** targetRow = 0; targetRow < targetImage.getHeight(); targetRow++) {  
 **for** (**int** targetCol = 0; targetCol < targetImage.getWidth(); targetCol++) {  
  
 **if** (targetImage.getMatrix().get(targetRow).get(targetCol) == sourceImage.getMatrix().get(row + targetRow).get(col + targetCol)) {  
 score++;  
 }  
 }  
 }  
 **return** score;  
}