Marathon Match

Problem Statement

Normal view

Contest: Robonaut Challenge (original contest)
Problem: RobonautEye

Problem Statement

Additional Information

In addition to reading the entire problem statement, please check the information provided in the following thread.

Project Overview

Robonaut 2 is the first humanoid robot in space and it was sent to the space station with the intention of taking over tasks too dangerous or too mundane for astronauts. But Robonaut 2 needs to learn how to interact with the types of input devices the astronauts use on the space station. To do that, we have built several taskboards for Robonaut to play with. The taskboards have a number of LEDs that turn on when the power switch is flipped, the buttons are pushed, and so on (see a video of Robonaut 2 interacting with a taskboard). In these challenges, you will control Robonaut and teach him how to interact with the taskboards.

Problem Statement

Your first challenge is to teach Robonaut how to recognize several buttons' and switches' state and location on the taskboard. To do this, you will be given a set of imagery from Robonaut both here on Earth, on the space station and in the simulator. Different kinds of images may differ quite significantly. Your algorithm must work for all of them. Though a strong performance on the real imagery is preferable over simulated imagery. You will be judged on accuracy (how often you detect state correctly and how close from actual position is your computed position) and runtime.

1 Competition Overview

Please note that this event is being run for a TopCoder customer. By winning a prize in this event you agree to transfer ownership of your submission to TopCoder.

This is a summary with highlights of the contest details. Please make sure to read all of the rules carefully before competing.

You must be eligible to compete in the match as defined in the rules.

This Marathon Match is a money match that has a **$10,000 prize purse!** The top 4 highest scorers will receive prizes with the first place winner receiving $4,000!

In TopCoder Marathon Match events, the system will be testing submissions for optimization according to the scoring criteria outlined in the problem statement that is associated with this event (this document). Each event may have a different scoring mechanism. Competitors will have to understand the scoring mechanism in order to effectively compete. *Participants may only submit code written in C++.*

Once the submission phase of a Marathon Match has ended, final testing will run and may take several days to complete. Upon completion of final testing, the final results will become available in the Marathon Match Archive on the web site. Competitors will be able to see the test case details, other competitors' submissions, and all final results. For rated events, the ratings will be adjusted once results are final. For more details on the Marathon Match format, click here.

2 Prizes

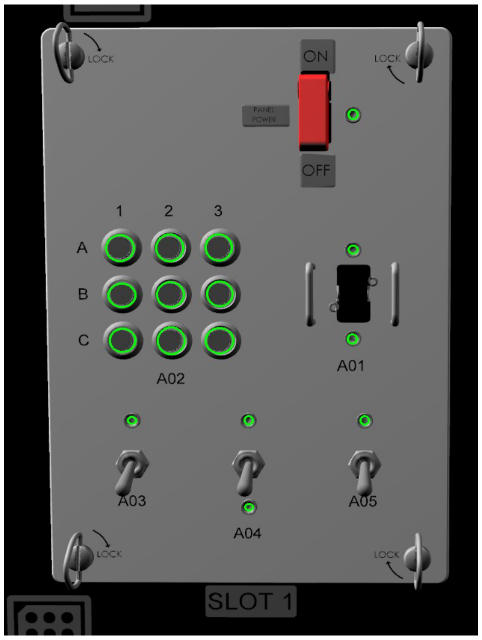
The **$10,000 prize purse** breaks down as follows:

```
1. $4,000
2. $3,000
3. $2,000
4. $1,000
```

TopCoder may *offer* to purchase submissions outside the top 4 if the client is interested in using them.

3 Taskboard details

This is what the taskboard looks like:



This file (XLS or CSV) contains the complete list of objects on the taskboard with their possible states and description.

(OUT object state will not be covered in the competition.)

This image illustrates a taskboard with all objects shown by blue circles. The numbers near circles are 0-based object numbers in the list of all objects. Object 5 (A01_ROCKER_LED_BOTTOM) is a LED a bit below object 3. It is not visible on this image and therefore no circle is drawn for it.

4 Implementation

For each test case you will be given two images – a “left eye” image and a “right eye” image. In your return you have to define the objects’ state and (x,y) location in pixels relative to the upper left corner of each image (“left eye” and “right eye”). The return value should contain 22 elements. Each element should correspond to a single object. The order of objects should be the same as in CSV/XLS file provided in the section "3 Taskboard details". Each elements should containing the following comma-separated parts: the object’s state, x coordinate in "left eye" image, y coordinate in "left eye" image, x coordinate in "right eye" image, y coordinate in "right eye" image. For example: "ON,125,120,53,63" (quotes for clarity only). X axis goes from left to right, Y axis goes from top to bottom, both coordinates are 0-based.

When an object is hidden or covered by an obstacle on both images, its state can't be identified even by a human. In this case the answer file will contain "HIDDEN" as its state and any state and locations will be considered correct in your return value.

When an object is hidden or covered by an obstacle only in one of the images (left or right), the answer file will contain "-1,-1" as its position for that image. In this case any location will be considered correct in your return value for that image (left or right).

Time limit is 1 minute per test case, but please consider that time is taken into account in scoring.

The input images are given by int[]s **leftEyeImage** and **rightEyeImage**. The format of each of them is as follows:

```
Height H (in pixels)
Width W (in pixels)
Data of pixel at row 0, column 0
Data of pixel at row 0, column 1
...
Data of pixel at row 0, column W-1
Data of pixel at row 1, column 0
...
Data of pixel at row 1, column W-1
...
...
Data of pixel at row H-1, column 0
...
Data of pixel at row H-1, column W-1
```

Rows are numbered from top to bottom. Columns are numbered from left to right. The data of each pixel is a single number calculated as 2^16 * Red + 2^8 * Green + Blue, where Red, Green and Blue are 8-bit RGB components of this pixel.

5 Testing and scoring

Train data:

You will get 126 pair of images that you can use to develop your algorithm. For these images we will also provide the expected (model) answers. All this data can be downloaded here.

Test cases:

All test data can be divided into 5 groups:

Group code	Origin	Images in training set	Images in provisional set	Images in system set
ISS	ISS	5	0	5
Lab	Earth	40	0	42
Lab2	Earth	34	27	35
Lab3	Earth	27	13	28
Sim	Simulator	20	10	20

The data was split between three test sets with a combination of random and manual approaches. The data contains some "clusters" -- huge groups of images that share the same Robonaut's position and look angle and also have a very similar lighting conditions. In almost all cases, the entire such "cluster" is kept within the same test set.

Scoring:

First of all, you will get a score of -1 for a test case if we were unable to get a return from your solution (due to time limit, memory limit, crashing, incorrect number of return or not properly formatted return). Assuming your return is proper, it is scored as 1,000,000 * correctness score * performance score * environment.

Environment score is the easiest to calculate. It is .5 for Sim, .7 for Lab3, .8 for Lab2, 1.0 for Lab and 1.1 for ISS images. The environment type is hidden from your method.

The **performance score** is calculated as 0.9 + 0.1 * ((5/max{T, 5})^0.5), where T is the runtime of your solution in seconds. So for example, the runtime less than or equals to 5 seconds results in 1.0 for performance score and runtime of 10 seconds results in ~0.9707.

Correctness score will be calculated by the following code:

```
function CorrectnessScore
BEGIN
  score := 0
  foreach object score = score + CorrectnessScoreSingle()
  RETURN score / 22
END.
```

```
function CorrectnessScoreSingle
BEGIN
  score := 0;
  IF realState <> “HIDDEN” THEN
  begin
    IF realLeftX = -1 AND realLeftY = -1 THEN score = score + 1
    ELSE
    begin
      distanceLeft := Sqrt( ( realLeftX – yourLeftX )^2 + ( realLeftY – yourLeftY )^2 )
      IF distanceLeft <= 2 * threshold THEN
      begin
        IF distanceLeft <= 1 * threshold THEN score = score + 1
        ELSE score = score + 1 - ( ((distanceLeft - threshold) / threshold) ^ 1.5 )
      end
    end

    IF realRightX = -1 AND realRightY = -1 THEN score = score + 1
    ELSE
    begin
      distanceRight := Sqrt( ( realRightX – yourRightX )^2 + ( realRightY – yourRightY )^2 )
      IF distanceRight <= 2 * threshold THEN
      begin
        IF distanceRight <= 1 * threshold THEN score = score + 1
        ELSE score = score + 1 - ( ((distanceRight - threshold) / threshold) ^ 1.5 )
      end
    end

    IF yourState = realState THEN score = score + 1
  end
  ELSE score = score + 3

  RETURN score / 3.0
END.
```

One of the reasons of having a threshold in scoring is that all "model" data has been produced manually and thus is not absolutely accurate.

Your overall score on a set of test cases is defined as the (sum of max{score, 0}) divided by (the sum of environment score). Both sums are calculated over all individual test cases from the set.

6 Tools

An offline tester tool is provided. It can run your solution locally, calculate its correctness score and has some simple visualization capabilities. It can also convert images to int[]s (according to the format used by **leftEyeImage** and **leftEyeImage**) and save the result to file.

We also provide the Java source code of the offline tester. You are allowed to introduce any modifications into the code in order to make it suit your needs better.

7 Special Conditions

In order to receive the prize money, you will need to submit a document explaining how your solution works. If you solved the task offline, meaning that your submission includes parameters, constants, or data generated off-line or prior to running your program, you will need to document the entire process you used to find the answers you submitted. That should include all code, scripts, and data you used in this process. Note that this information should not be submitted anywhere during the coding phase. Instead, if you win a prize, a TopCoder representative will contact you directly in order to collect this data. You will need to submit all the required information within 7 days after the contest results are published. Questions sent by email from TopCoder requesting clarification to this document must be answered within 3 days. If the document or a response to a clarification request is not received, TopCoder reserves the right to disqualify the submission.

Definition

Class:	RobonautEye
Method:	recognizeObjects
Parameters:	int[], int[]
Returns:	String[]
Method signature:	String[] recognizeObjects(int[] leftEyeImage, int[] rightEyeImage)

(be sure your method is public)

Notes

- Time limit is 1 minute per test case and memory limit is 1024MB.
- There is no explicit code size limit. The implicit source code size limit is around 1 MB (it is not advisable to submit codes of size close to that or larger). Once your code is compiled, the binary size should not exceed 1 MB.
- The compilation time limit is 30 seconds. You can find information about compilers that we use and compilation options here.
- You can include open source code in your submission provided that it is licensed under an OSI-approved open source license, the open source code is separated from and clearly identified in your code, and your submission in this competition is in compliance with the license.
- *No usage of assembly is allowed.*
- The match forum is located here. Please visit it regularly during the match, since we may post updates and/or clarifications there.

Constraints

- **Threshold** is 10 for Sim, Lab2 and Lab3 images, and 15 for Lab and ISS images.
- **Width** x **Height** is 2448 x 2050 for Lab and ISS, and 1600 x 1200 for Sim, Lab2 and Lab3 images.
- **Red**, **Green**, **Blue** will be between 0 and 255 (inclusive), thus **Data of pixel** will be between 0 (inclusive) and 2^24 (exclusive).

Examples

0)

```
Folder: ISS
Image: BHXG.tif
Threshold: 15
```

1)

```
Folder: ISS
Image: CIZA.tif
Threshold: 15
```

2)

```
Folder: Lab
Image: AHJH
Threshold: 15
```

3)

```
Folder: Lab
Image: AJBB
Threshold: 15
```

4)

```
Folder: Lab2
Image: ALJD
Threshold: 10
```

5)

```
Folder: Lab2
Image: AMHF
Threshold: 10
```

6)

```
Folder: Lab3
Image: AIFS
Threshold: 10
```

7)

```
Folder: Lab3
Image: AWZM
Threshold: 10
```

8)

```
Folder: Sim
Image: BGML
Threshold: 10
```

9)

```
Folder: Sim
Image: CRSP
Threshold: 10
```