Find The People: A Lisp Programming Contest *

TC Lispers
http://tclispers.org/

July 31, 2010

Abstract

Your Lisp program will be given two input images from the same web cam. The first image will contain zero people, the second image will contain at least one person.

Your program's goal is to find the people. You will get points for emitting the pixel coordinates of a pixel inside a person. You'll get bonus points if it's a head pixel. Be careful, though. You'll lose points for emitting more than one pixel per person.

Deadline: September 19th.

Target Platform: SBCL or Clozure under Mac OS X (10.6.4).

Prizes for: Tournament winner, Prettiest code

Web Page: `http://tclispers.org/contest-find-the-people`

1 Input

The pictures will be taken from this web cam's gallery (*note:* please, do not slam their site trying to download ten zillion practice pictures... take a reasonable number of hand-selected images...):

`http://www.bucuticam.com/album2/gallery.php`

The pictures are 640 pixels wide and 480 pixels high. They will be presented to you as 480 x 640 x 3 Lisp arrays. These arrays will be available in readable Lisp forms in the file "`in.lisp`" in the current directory and on `*STANDARD-INPUT*`. Each element of the array will be a floating point number between zero and one (inclusive). The first array will be a reference image containing no people. The second image will be an image from the same day which contains at least one person.

An example input is available here:

`http://tinyurl.com/find-people-example-lisp`

To illustrate, this is what an image might look like if it were a 4 x 6 x 3 Lisp array rather than a 480 x 640 x 3 array:

*Patrick Stein pat@nklein.com, Robert Goldman rpgoldman@sift.info

```
#3A(((1.0 0.0 0.0) (0.5 0.5 0.0) (0.0 1.0 0.0) (0.0 0.5 0.5)
     (0.0 0.0 1.0) (0.5 0.0 0.5))
    ((1.0 1.0 0.0) (0.8 1.0 0.8) (0.0 1.0 1.0) (1.0 0.8 0.8)
     (1.0 0.0 1.0) (0.8 0.8 1.0))
    ((0.0 0.0 0.0) (0.2 0.2 0.2) (0.4 0.4 0.4) (0.6 0.6 0.6)
     (0.8 0.8 0.8) (1.0 1.0 1.0))
    ((1.0 0.0 0.5) (0.8 0.2 0.5) (0.6 0.4 0.5) (0.4 0.2 0.5)
     (0.2 0.8 0.5) (0.0 1.0 0.5)))
```

Each three-element subarray represents one pixel. The first number in that pixel is its red component. The second number in that pixel is its green component. The third number in that pixel is its blue component.

The image is represented LRTB: Left-to-Right, Top-to-Bottom. This means that the first element of the array corresponds to the top-left corner of the input image. The last element of the array corresponds to the lower-right corner of the input image. Further, if you stepped through in `ROW-MAJOR-INDEX` order, you would be going through each color component of a pixel before moving on to the next pixel and horizontally across each pixel in a line before continuing on to the next line.

Here is an example reference image:

`http://www.bucuticam.com/album2/Images/6.zoom2.jpg`

Here is an example target image:

`http://www.bucuticam.com/album2/Images/6.zoom1.jpg`

2 Output

Your program is to output a series of two-element lists of pixel coordinates. The pixel coordinates are the first and second index of the pixel in the input array. For example, the first pixel in the array (which is "`(1.0 0.0 0.0)`" in the example shown above) is $(0, 0)$. To declare that that pixel is inside a person, your program should output "`(0 0)`". To claim that all four corner pixels in the image are "people pixels", your program should output (in any order):

```
(0 0) (0 639) (479 0) (479 639)
```

The parentheses are required. Whitespace is required between the first and second number of each coordinate. All other whitespace is optional.

3 Scoring

Your program will receive 100 points for each person it discovers. You program will receive 50 bonus points if at least one pixel your program found was in the person's head.

Your program will lose 15 points for the second, third, fourth, and fifth pixel found within a given person. Your program will lose 40 points for each pixel beyond the fifth found within a given person.

Your program will lose 10 points for the first, second, third, fourth, and fifth pixels not within any person. Your program will lose 35 points for each pixel beyond the fifth that is not within any person.

For example, say the image contains Alice, Bob, and Carol. Say, my program outputs the following pixels:

Two in Alice's torso
Two in Bob's torso
Four in Bob's head
Nine in the sky.

My program would receive:

+100	for finding Alice
-15	for hitting Alice the second time
+100	for finding Bob
+50	for hitting Bob's head
-15	for hitting Bob the second time
-15	for hitting Bob the third time
-15	for hitting Bob the fourth time
-15	for hitting Bob the fifth time
-40	for hitting Bob the sixth time
-10	for hitting non-people the first time
-10	for hitting non-people the second time
-10	for hitting non-people the third time
-10	for hitting non-people the fourth time
-10	for hitting non-people the fifth time
-35	for hitting non-people the sixth time
-35	for hitting non-people the seventh time
-35	for hitting non-people the eighth time
-35	for hitting non-people the ninth time

Overall, my program would have -140 points.

Your program, being much more frugal, output only:

One in Alice's head
One in Carol's head

Your program would receive:

+100	for hitting Alice
+50	for hitting Alice's head
+100	for hitting Carol
+50	for hitting Carol's head

You program would have a total of 300 points.

If Paul's program were even more frugal and kept totally silent, it would have scored zero points putting it ahead of my program and behind your program.

4 Some Details About Scoring

There is no penalty for not noticing a person.

It does not matter what order you output your coordinate pairs. You will always get credit for the first head pixel on a given person even if you've already emitted a body pixel for that person.

The classification of each pixel as either head or person or not will be done manually before an image is tested against any program. Thus, the decisions will not be influenced by what any particular program thinks.

Pixels will be considered head pixels if they are head, hair, hat, or sunglasses. A hand or sign in front of the face will not be considered a head pixel.

Pixels will be considered body pixels if they are not head pixels but are clothing, skin, or an item held in the person's hand (such as a beer can or portable sign).

If a person is partly obscured by the trees or the permanent sign in the scene or chairs or bikes or other such things, the obscuring pixels will NOT be considered body pixels. Those pixels will be considered an erroneous guess and will incur the penalties for guessing a pixel not within a body.

If a pixel is part of the border between one person and another in the image, that pixel will be entirely ignored in judging. It will not score points for hitting a person. It will not incur penalties for missing the person. It is as if you never guessed it.

Here is a manual classification of the pixels in the sample image linked to above:

`http://tinyurl.com/find-people-example-map-png`

Each body pixel has full-on red and zero green. Each head pixel has full-on red and full-on green. Each person has a different value for the blue channel. Pixels that are a miss altogether are black. Pixels that are on the border between two people (and will be ignored) have half-on red, green, and blue channels.

5 Tournament

For dramatic effect, rather that running each program against the test images and picking the highest total score, the programs will be competing in a tournament format.

Each round of the tournament will consist of one reference image and one target image. Every program will be scored on the image pair for the round. The worst scoring half of the programs will be eliminated from the tournament until there are just three contestants remaining.

Technical note: the first round will not necessarily eliminate half of the programs. Rather, it will eliminate as many programs as needed to get the number of remaining contestants down to a three times a power of two. So, given `NN` participants with `NN` greater than three, the first round will eliminate this many contestants:

```
(- NN (* 3 (EXPT 2 (CEILING (LOG (/ NN 6) 2)))))
```

Those three will compete on a final image with their placing determined by their score on this final image.

To facilitate the scoring, any program that has not made its first guess within five minutes or has gone longer than a minute between guesses or has been running for longer than twenty minutes will be terminated with whatever score it has at the time.

6 Code Judging Criteria

A panel of judges will receive an anonymized copy of the source code you wrote for this contest. Each judge will score your code in the following categories: Elegance, Maintainability, and Lispiness. Each judge can award you up to twenty points in each category.

The "Elegance" category is meant to embody how your code fits together as a whole, how it sits, how it flows, how it's crafted, etc.

The "Maintainability" category is meant to cover how easily a moderately skilled Lisp coder could adapt your code to some new purpose, find a bug in your code (obviously introduced by some third party because your code was bug-free), port the code to a new Lisp implementation or Lisp-like language, etc.

The "Lispiness" category is meant to embody things like how idiomatic your code is (and how well you choose when to break with idiomatic usage), how your code demonstrates a mastery of the breadth of the Common Lisp standard (you didn't reimplement `COUNT-IF`, did you?), etc.

Your score in each category will be the average of the scores given by the judges after your top and bottom score in that category are eliminated. Your final score will be the sum of your score in the three categories.

7 Prizes

We are working to procure prizes for the top finishers. Once those are finalized, they will be announced on the contest web page:

`http://tclispers.org/contest-find-the-people`

8 Environment

Your submission will be untarred or unzipped in a directory of its own. It will get a copy of the "`in.lisp`" file in the top-level directory for your submission. The "`in.lisp`" file will also be given to your program on `*STANDARD-INPUT*`.

In the top-level directory of your submission, you should have a file called "`run-sbcl.lisp`" or "`run-ccl.lisp`". Here is what we will do (effectively) to run your program:

```
# get to your directory
```

```
% cd /path/to/your/top-level/submission/directory

       # copy the in.lisp file to this directory
% cp /the/untouchable/real/original/in.lisp ./in.lisp

       # run your program (assuming you have a run-sbcl.lisp)
% sbcl --no-sysinit --no-userinit --load run-sbcl.lisp < in.lisp

       # run your program (assuming you have a run-ccl.lisp)
% ccl --no-init --load run-ccl.lisp < in.lisp
```

Your program will be judged on the output written to `*STANDARD-OUTPUT*`. All other output will be ignored. Your program will be disqualified if it opens any windows of its own or tries to do that is, in the judges' view, malicious or potentially harmful to the judging machine or its contents (including, but not limited to, reading files outside of its own directory, creating inordinately large files, using the network, creating background programs that persist longer than the tournament run, executing programs outside of your directory, etc.).

You can count on being able to `(REQUIRE :ASDF)`. You cannot count on any other packages being available beyond those available by default with SBCL or Clozure (obviously, you cannot depend on Clozure's stuff from SBCL or vice-versa). You are free to use any package you so desire, but you are responsible for including it in your submission and getting it loaded (i.e. setting up the `ASDF:*CENTRAL-REGISTRY*` as needed and loading the packages).

9 Submissions

Your submission should be in the form of a zip file, tar file, or compressed (.Z, .gz, or .bz2) tar file. Make this tar file available somewhere on the net and post a comment on the contest web page declaring the location of your entry. The contest web page is here:

`http://tclispers.org/contest-find-the-people`

If you are unable to do this, please email pat@nklein.com to arrange an alternate submission process.

No submissions will be accepted after 23:59:59 UTC on September 19th.

To facilitate judging of code, please include a one-line comment as the first line of each source file that you wrote stating:

```
;;; original contest code by YOUR-NAME-AND-EMAIL-ADDRESS-HERE
```

(obviously, with your name and email address in the indicated spot). Also, please do not include your name, email address, or identifying information anywhere else in your source code.