Topcoder - Cassini Phase 1 Marathon Match

'*andresduque*' Solution description

Summary

All of these steps are done for each passed image to **testingData** method except step 6 which is done after all images where processed and **getAnswer** method is called. Passed data to **trainingData** method is not used.

1. Input image edges seems to have invalid pixels values, so all 4 edges pixels are set to closest pixel value.
2. Process the input image to get an output image that contains objects that are possible propellers, although also can be rings borders, image artifacts, or other type of objects.
3. Find a threshold value that can binarize output image.
4. Extract objects finding blocks of 4 connected pixels that are greater than the threshold found at step 3. If a found object is too big or too small the object is ignored and not added to the found list.
5. All found objects get a score based on the contrast of the region the object is located and the list of objects is ordered based on this score. The idea is that propellers are generally located in low contrast areas, so objects with low contrast area are put first in the list.
6. Finally after all images where processed (and **getAnswer** method is called), some objects are removed using 2 different methods. This step try to maximize score removing objects that probably are not propellers.

Details

Note: All objects found in each image get the same id, at each call of **testingData** method the global id counter is increased.

1. **borderClone** method

   Used to set all edges pixels to a closest pixel value that is not in an edge.

2. **cleanRings** method

The idea is to process the image in square sub images (sliding window technique), for each sub image the algorithm find square image regions that don't intersect with current sub image and calculate a score. These regions are of the same dimension as current sub image. Score for each region is simply the absolute difference of each of the pixels in the region with each of the pixels in the sub image.

The pixels in the output image that correspond to the current sub image are calculated using the region that get the lowest score of all tested regions. Each pixel is set to the absolute difference of sub image pixel and region pixel.

Algorithm continue processing all sub windows and complete the output image.

Notes: Algorithm seems good as resulting output image emphasize strange objects that can be propellers. Still output image contain accentuated areas that are generally the rings borders and this cause false positives in the next steps. Output image could be further processed to reduce noise which also increase false positives. Also sub image size can have an effect in solution performance.

3. **findThresholds** method

   Starting value is the pixel with highest intensity, we scan all image pixels and count number of pixels that are greater than current value. At each iteration we reduce current value. We continue until:

   *(count / (size * size) > 0.003)* Where size is size of image.

   Notes: Idea is just to find a good enough value so in next step extracted objects number is not too big. Alternative methods to calculate threshold value could improve detection result.

4. **Extract.extract** method

   Algorithm find 4 connected components of pixels that are greater than passed threshold (calculated at step 3)

   All found components that are not too small or too big are used to create objects. Each object position, dimension and center point are set. The id of all found objects for the current image is set to the same value.

   Finally the list with all found objects is returned.

5. **contrastScore** method

   All of the objects found at step 4, are scored based on the contrast of the area of the object. The bounding box of the object plus a padding is used as the region to calculate the contrast.

   After that, all objects are ordered in ascending order using the contrast score.

Note: This generally help improve score, but in some cases propellers can be in high contrast regions.

6. Two methods are called to remove objects that possible are not propellers and to link objects that are probably the same object in different images. This is done after all images where processed and **getAnswer** method is called.

- **similarObjects** method
    - Idea is to find group of objects present in multiple images at close locations and remove them. When we found a group of objects present in 2 images, the images are probably similar and of the same area of the planet, so also we set the ids of objects the same (because found propellers are probably the same object). Because propellers move in each image, they are hopefully not removed and only other objects or image artifacts are removed.
    - Algorithm find at each processed image objects that are also present in other image. Objects positions must be close but not exact (tolerance variable used). If there are at least 15 objects, all of them are removed from current list of objects for that image. Also to improve link scoring, method change all ids of all objects in current image to the id used in the matched image.
- **removeNoiseObjects** method
    - Idea is to remove groups of close objects which can be image noise and not real objects. For each image algorithm find group of objects that are close enough (based on tolerance variable). All objects that are part of a group of at least 20 objects are removed.
- Note: Above two methods help increase score, improving code can probably increase score of solution.