Topcoder Marathon Match – Propeller Hunt 1
October/2014
Solution Description
Wladimir Leite (wleite)
Final Score: 232,300.25 (1st place)

TRAINING PHASE: For each new training image, it processes the image and extract features of ground truth (correct propellers detections) and other random points (not propellers).

1. **Remove rings**: process the input image using the provided `RingSubtractor` implementation. The only change was the value of `terminateN` to reduce the maximum number of cycles from 3000 to 200. This reduces a lot the time spent on this step. With the original value it would be impossible to process all images in the given time of 60 minutes. Other minor loop optimizations were made, using temporary variables when accessing bi-dimensional arrays inside nested loops.

2. **Convert the image to "radial" coordinate system:** The original input image has the original point of view coordinate system, but the relative position of the propeller and the rings are very different across the images. To make images more similar, a conversion between the original coordinate system and a new system where the X-axis is the longitude and Y-axis is the radius. The following figures show this transformation (which uses provided `radiusLongitude2PixelPosition` method of `Transformer` class to make the necessary calculations). After this transformation the remaining rings artifacts and the propellers look "more horizontal".

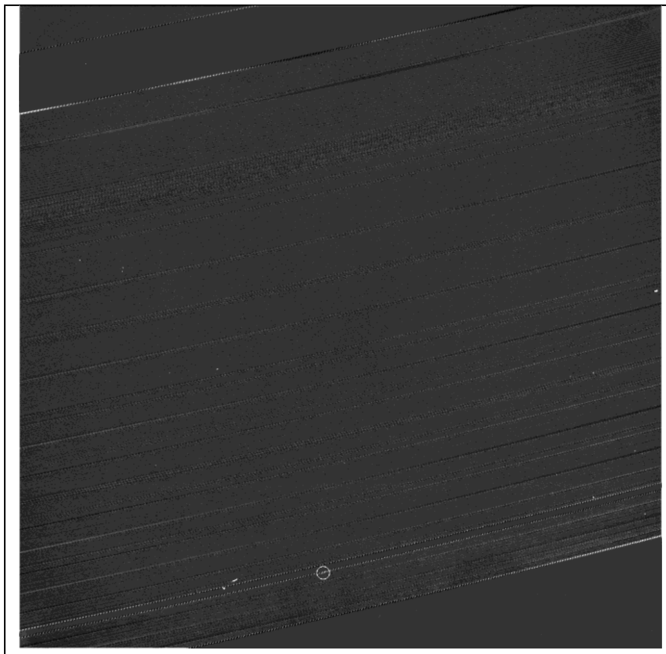
Figure 01 - Original image, after RingSubtractor step, with a circle around the known propeller.

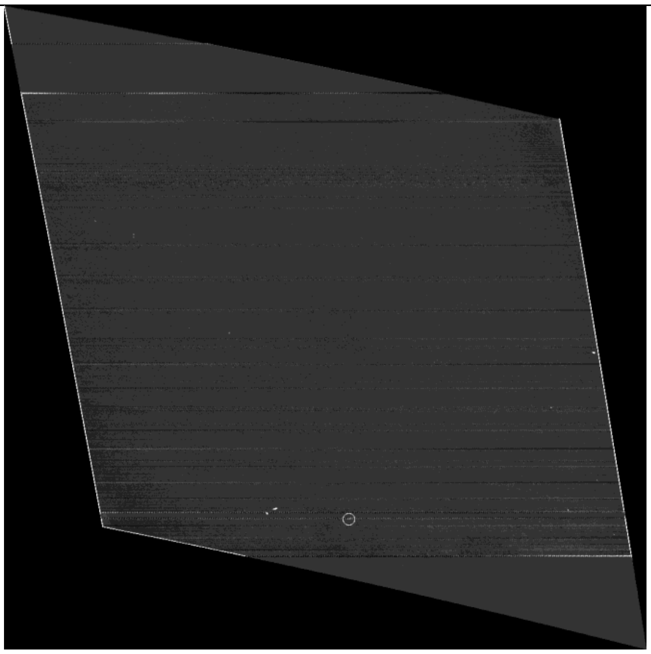
Figure 02 – The same image, after the transformation to "radial" coordinates, where X-axis is the longitude and Y-axis is the radius.

3. **Convert ground truth coordinates to "radial" coordinate system:** Using the same transformation, the given coordinates of known propellers are converted to the new coordinate system.

4. **Extract features of ground truth points**: Instead of finding connected components which would be a "standard" approach, I tried considering only the neighborhood of a given point. I didn't have time to implement a method to find connect component and use its size and dimensions as features, but it seems that would help to improve score. The extract features are the statistical values (mean, variance and skewness) of rectangles which center is the given point. The used rectangle are (in pixels, although I didn't tried, maybe using fixed radius and longitude units could give best results):
    - (x – 1, y – 1, x + 1, y + 1) : (Left, Top, Bottom, Right)
    - (x – 3, y – 3, x + 3, y + 3)
    - (x – 5, y – 5, x + 5, y + 5)
    - (x – 7, y – 7, x + 7, y + 7)
    - (x – 8, y – 4, x + 8, y + 4)
    - (x – 10, y – 4, x + 10, y + 4)
    - (x – 12, y – 4, x + 12, y + 4)

    Other features that I found very useful are the difference between each of the three statistical values of rectangles centered in the given point and the four adjacent rectangles of the same size. These differences are calculated for the following rectangles:
    - (x – 10, y – 4, x + 10, y + 4)
    - (x – 6, y – 3, x + 6, y + 3)

5. **Extract features of other (not propellers) points**: Using the same method described in the previous item, features are extracted from other 15,000 random points of the image.

6. **Random forest building**: Using the extracted features of correct (propellers) and incorrect detections (random points) it builds a random forest of 256 classification trees.

TESTING PHASE: For each testing image, it processes the image the same way described before (steps 1 and 2). Then it extracts features (as described in step 4) of "many" points (every 5 in X-axis and every 3 in Y-axis). After the end of this phase I found out that reducing the sampling distance (from 3 and 2, for example) would give better results, although the necessary processing time is increased. Here the inverse transformation is used to find the pixel coordinate in the original image of a given point in the transformed (radial) image. Finally it uses the trained random forest to predict the chance of a given point be a propeller.

ANSWERING PHASE: It simply orders detections by decreasing probability of being a propeller. The assigned group, due to lack of time to investigate other approaches, is simply the integer division between the point radius and a constant value of 500.