## To be done immediately: [April 28, 2011]

Prepare training matrices from labeled regions. Error checking for trained labels: duplicate regids, non-existent regids.

Finalize the architecture / design of the project. Commit to git hub.

## Data Collection Sources:

1. Manually clicking birds using a camera-enabled phone
2. MediaDownloader
3. ImageNet

## Current Mechanisms:

### Preprocessing:

Resize the input image such that width = 400 pixels. Typically this will result in a 400x300 image, since cell phone cameras and most other consumer cameras capture images in the 4:3 aspect ratio.



### Segmentation:

Region growing based on color. The (empirical) optimum threshold is 0.10. Other alternatives are Ncuts, clustering, etc. but region growing is intuitive to grow highly homogeneous regions and simple enough to code by hand if required to be done when porting to a mobile device. Owing to the cleanup that will be performed, a smaller threshold can be used for segmentation.

Code was downloaded for region growing (based on grayscale intensity) from the internet. This was enhanced to grow regions considering color. The algorithm was iteratively applied over the entire image until all pixels belonged to exactly one region. The result is as below:



### Subsuming Regions:

As seen in the above region, there are a huge number of regions, most of them being very tiny (a few square pixels). Hence it is desired to cleanup these regions. Below are the steps taken to do this:

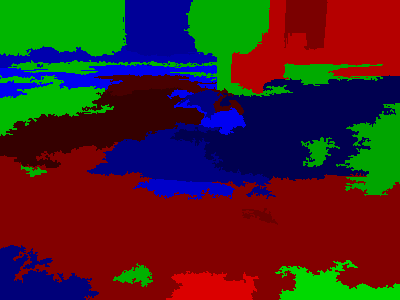
Stage 1: Clean up the really tiny regions (less than 65 square pixels) by looking inside a window around such regions, picking the region whose pixels are the most and merging to that region. This gets rid of a lot of “dirt” in the segmented output.

**

[OBSOLETE] Stage 2: Look for “island” regions (less than 200 square pixels) and merge them into the “sea”. A region qualifies to be an island if all its outer contour pixels belong to exactly one other region. The outer contour of a region is found by calculating the inner contour using a library. The intuition here is that a bird body part is never going to be a dangling blob and hence it makes sense to merge it into the background.



[OBSOLETE] Stage 3: Clean up the regions that are relatively small (less than 1000 square pixels) by merging them to their nearest neighbor. This makes regions bigger. In this step, the biggest 3 regions are never merged into; the intuition being that such regions will probably be well established backgrounds like grass, roads, mud, snow, water, etc. Hence the mid-sized regions grow in this step. As of now, the implementation considers all regions inside a certain window to qualify as a neighbor, hence a neighbor could be a region that is in the vicinity but does not share a border with the target region.



### Segmentation Output:

A partly random coloring scheme is implemented, which chooses random tints of red, blue and green in that order for coloring regions. By choosing all three channels in a loop, there is some control over the colors of adjacent regions so that they do not get colored with the same channel. However, due to the arbitrary nature of segments, some adjacent regions do end up having colors of different tints for the same channel. A safeguard is implemented to ensure that none of the segments are colored completely black.

### Saving the segmented image:

The parameters used for the processing are saved as part of the filename of the PNG image. These include the segmentation threshold, the size thresholds for small and island regions, the number of segments, the running time in seconds, etc. The output format is PNG.

For large enough regions, the region ID is shown near the leftmost pixel of the region in the PNG image. Printing characters is done using code downloaded from the internet. However this code was not usable directly owing to the large size of the font used. Hence this code was modified to print smaller characters in the output PNG image.

### Classifier Training and Testing:

The segments need to be identified. The segments can fall into any of the below categories:

1. Grass
2. Buildings
3. Mud
4. Roads
5. Snow
6. Water
7. Sky
8. Leaves
9. Bark
10. Sand
11. Other
12. Bird

A classifier (linear SVM) is trained to identify regions. The features used for identification are color histograms (64 bins for each channel = 192) and SIFT visual words (175).

Ideally, the classifier needs to be trained on labeled segments. However, the time required to segment a single image reduced to 400x300 pixels is about 3 to 5 minutes. Training segments alone is not feasible; hence (homogeneous) crops of images are used along with segments to train the classifier.

This task can be split into two sections:

Part 1: Recognize stuff like grass, snow, water, mud, roads and buildings which makes up the background of the image. Most of the image is expected to be made up of such stuff considering that cell phone cameras do not have optical zoom.

Ideally, the classifier should be trained on actual segments of images. But segmenting images takes up a lot of time, hence as a quick prototype solution, rectangular sections of stuff are saved manually to form a set of training images.

Color histograms + SIFT features are computed for these and a SVM will be trained to recognize these. The color histograms are computed using 64 bins for each channel, which results in 192 features. The SIFT features from all images are clustered into 175 visual words and a count of each visual word is computed for each image for training / testing. The counts in both the color histogram and SIFT visual words are normalized by dividing the counts by the number of pixels in the image (area).

An SVM is trained for each category with values 800 and 0.1 for c and g respectively.

Part 2: Recognize actual bird parts like a bird’s head, beak, etc. Perhaps using a trained object detector?

### Text Processing:

The complete eBird New York database is used which has around 5 million entries in total, 1.5 million are after January 2009 and these are the ones used in the analysis. An SQL query is able to extract a histogram of bird counts (bird name + observation counts) within a desired radius around a given GPS location. This requires computing the distance between two GPS locations in miles, and it is done using a dedicated stored function in MS SQL Server. Moreover, the month of the sighting is also taken into account, around which a delta of 1 month is applied while fetching the observation counts.

The histogram will provide probabilistic information.

Bugs: Wrap around for months required, count actual sightings rather than whole records, analyze how many records per month, average out if needed.

## Performance and Results:

Typical running time is 5 minutes to get to the (cleaned up) segmented stage.

The cleaning up of regions reduces the total number of regions from around 7000-26000 (depends on the image) to around 25-100.

## Future Work:

1. Training classifiers to recognize specific bird features like the head, beak, etc.
2. Extracting information like color / texture from those regions.

## Nice to have features:

1. Java naming convention
2. Optimize region grower, currently too slow
3. Better algorithm to subsume regions. E.g. make use of color information / gradients
4. Explore other ways of subsuming regions / segmentation (e.g. Watershed, Maire)
5. A coloring scheme for segmentation that takes into account the relative “closeness” of regions

## Hardware and Software:

A laptop machine with a Core 2 Duo processor with 2 GB RAM, using 32-bit MATLAB.

## Testing:

## Downloaded Code:

1. Grayscale intensity single region growing -- D. Kroon, University of Twente
2. Moore neighborhood (contour) of a region -- Adam H. Aitkenhead
3. Substring function for MATLAB – Peter J. Acklam, Phillip M. Feldman
4. Rendering text over images – Davide Di Gloria