WndCharm Notes

"Weighted Neighbor Distance using Compound Hierarchy of Algorithms Representing Morphology"

In CentOS 6.6 Linux box …

Dependencies to install

```
sudo yum install libtiff-devel
sudo yum install fftw-static fftw-devel
sudo yum install fftw-static fftw-devel
sudo yum install libX11-devel libXt-devel libXaw-devel
sudo yum install ImageMagick ImageMagick-devel
```

Download zip

https://github.com/wnd-charm/wnd-charm

Download zip directly to Linux box to preserve file permissions

```
[129 efg localhost 2015-02-04 21:08:43 /home/efg/Software/wnd-charm-master]
./configure
```

```
[130 efg localhost 2015-02-04 21:10:03 /home/efg/Software/wnd-charm-master]
make
```

```
[131 efg localhost 2015-02-04 21:10:48 /home/efg/Software/wnd-charm-master]
sudo make install
```

```
workon Python279
```

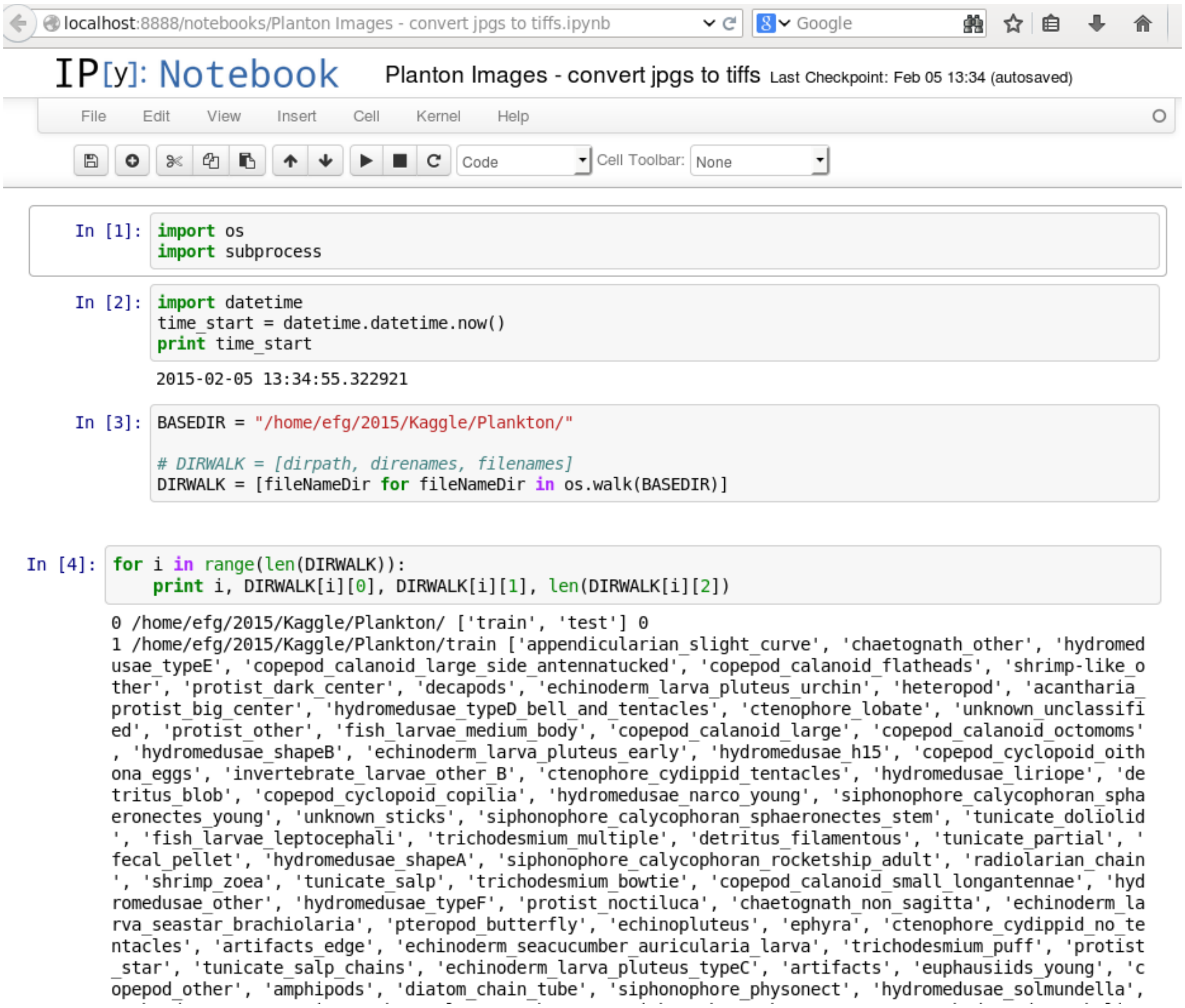
Convert files: .jpg to .tif

The *wndchrm* program only processes .tif images, so the .jpg files must be converted. *ImageMagick* will be used for this conversion: http://www.imagemagick.org/

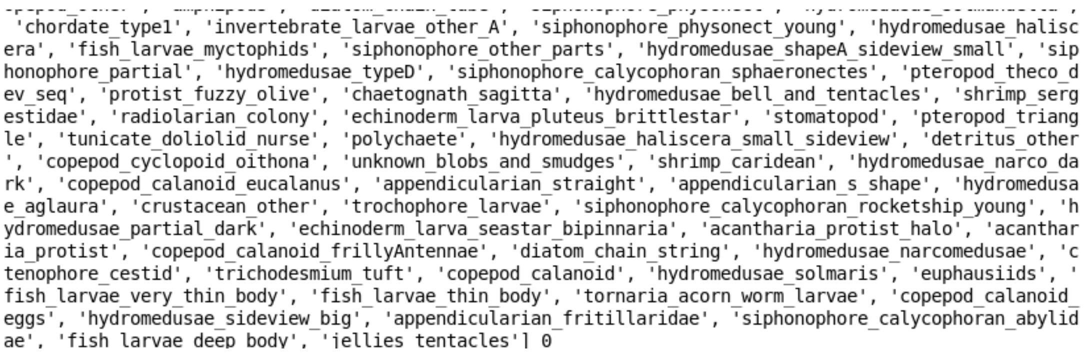
An IPython script was used to document the conversion.

```
[274 efg localhost 2015-02-05 17:28:21 /home/efg/2015/IPython]
ipython notebook
```

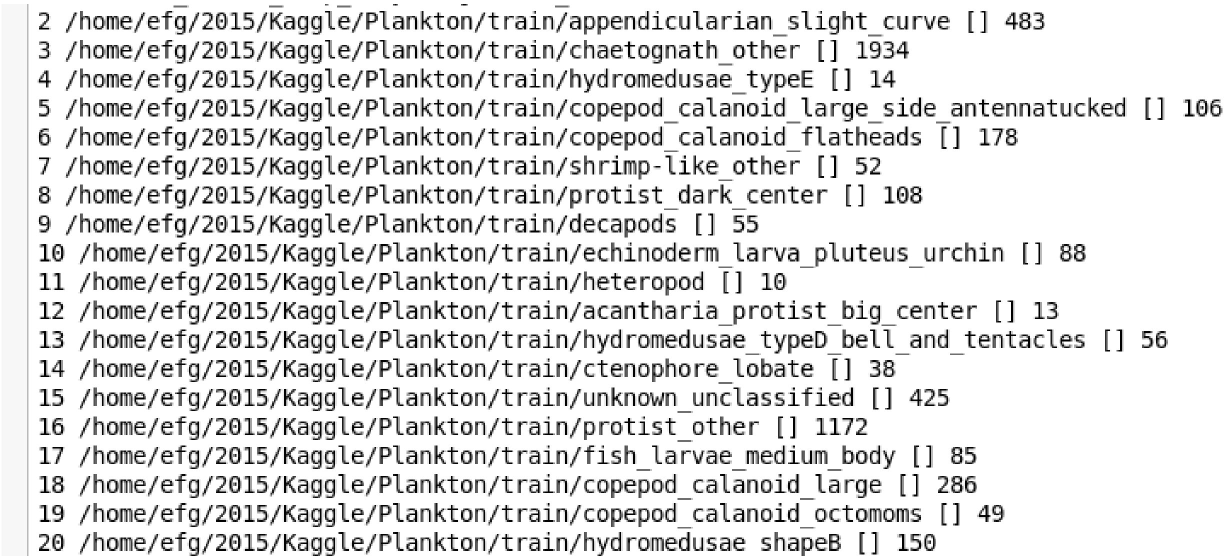
WndCharm Notes


localhost:8888/notebooks/Planton Images - convert jpgs to tiffs.ipynb
Google
IP[y]: Notebook
Planton Images - convert jpgs to tiffs Last Checkpoint: Feb 05 13:34 (autosaved)
File
Edit
View
Insert
Cell
Kernel
Help
Code
Cell Toolbar:
None

```
In [1]: import os
        import subprocess

In [2]: import datetime
        time_start = datetime.datetime.now()
        print time_start

        2015-02-05 13:34:55.322921

In [3]: BASEDIR = "/home/efg/2015/Kaggle/Plankton/"

        # DIRWALK = [dirpath, direnames, filenames]
        DIRWALK = [fileNameDir for fileNameDir in os.walk(BASEDIR)]


In [4]: for i in range(len(DIRWALK)):
            print i, DIRWALK[i][0], DIRWALK[i][1], len(DIRWALK[i][2])

        0 /home/efg/2015/Kaggle/Plankton/ ['train', 'test'] 0
        1 /home/efg/2015/Kaggle/Plankton/train ['appendicularian_slight_curve', 'chaetognath_other', 'hydromed
        usae_typeE', 'copepod_calanoid_large_side_antennatucked', 'copepod_calanoid_flatheads', 'shrimp-like_o
        ther', 'protist_dark_center', 'decapods', 'echinoderm_larva_pluteus_urchin', 'heteropod', 'acantharia_
        protist_big_center', 'hydromedusae_typeD_bell_and_tentacles', 'ctenophore_lobate', 'unknown_unclassifi
        ed', 'protist_other', 'fish_larvae_medium_body', 'copepod_calanoid_large', 'copepod_calanoid_octomoms'
        , 'hydromedusae_shapeB', 'echinoderm_larva_pluteus_early', 'hydromedusae_h15', 'copepod_cyclopoid_oith
        ona_eggs', 'invertebrate_larvae_other_B', 'ctenophore_cydippid_tentacles', 'hydromedusae_liriope', 'de
        tritus_blob', 'copepod_cyclopoid_copilia', 'hydromedusae_narco_young', 'siphonophore_calycophoran_spha
        eronectes_young', 'unknown_sticks', 'siphonophore_calycophoran_sphaeronectes_stem', 'tunicate_doliolid
        ', 'fish_larvae_leptocephali', 'trichodesmium_multiple', 'detritus_filamentous', 'tunicate_partial', '
        fecal_pellet', 'hydromedusae_shapeA', 'siphonophore_calycophoran_rocketship_adult', 'radiolarian_chain
        ', 'shrimp_zoea', 'tunicate_salp', 'trichodesmium_bowtie', 'copepod_calanoid_small_longantennae', 'hyd
        romedusae_other', 'hydromedusae_typeF', 'protist_noctiluca', 'chaetognath_non_sagitta', 'echinoderm_la
        rva_seastar_brachiolaria', 'pteropod_butterfly', 'echinopluteus', 'ephyra', 'ctenophore_cydippid_no_te
        ntacles', 'artifacts_edge', 'echinoderm_seacucumber_auricularia_larva', 'trichodesmium_puff', 'protist
        _star', 'tunicate_salp_chains', 'echinoderm_larva_pluteus_typeC', 'artifacts', 'euphausiids_young', 'c
        opepod_other', 'amphipods', 'diatom_chain_tube', 'siphonophore_physonect', 'hydromedusae_solmundella',
```

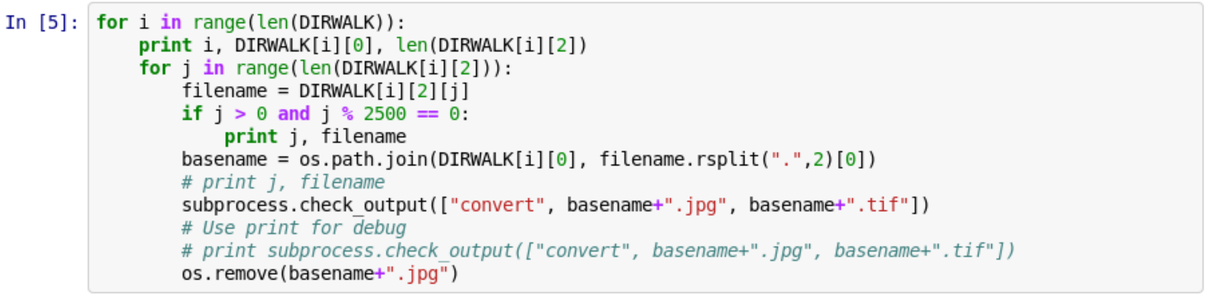


```
 'chordate_type1', 'invertebrate_larvae_other_A', 'siphonophore_physonect_young', 'hydromedusae_halisc
era', 'fish_larvae_myctophids', 'siphonophore_other_parts', 'hydromedusae_shapeA_sideview_small', 'sip
honophore_partial', 'hydromedusae_typeD', 'siphonophore_calycophoran_sphaeronectes', 'pteropod_theco_d
ev_seq', 'protist_fuzzy_olive', 'chaetognath_sagitta', 'hydromedusae_bell_and_tentacles', 'shrimp_serg
estidae', 'radiolarian_colony', 'echinoderm_larva_pluteus_brittlestar', 'stomatopod', 'pteropod_triang
le', 'tunicate_doliolid_nurse', 'polychaete', 'hydromedusae_haliscera_small_sideview', 'detritus_other
', 'copepod_cyclopoid_oithona', 'unknown_blobs_and_smudges', 'shrimp_caridean', 'hydromedusae_narco_da
rk', 'copepod_calanoid_eucalanus', 'appendicularian_straight', 'appendicularian_s_shape', 'hydromedusa
e_aglaura', 'crustacean_other', 'trochophore_larvae', 'siphonophore_calycophoran_rocketship_young', 'h
ydromedusae_partial_dark', 'echinoderm_larva_seastar_bipinnaria', 'acantharia_protist_halo', 'acanthar
ia_protist', 'copepod_calanoid_frillyAntennae', 'diatom_chain_string', 'hydromedusae_narcomedusae', 'c
tenophore_cestid', 'trichodesmium_tuft', 'copepod_calanoid', 'hydromedusae_solmaris', 'euphausiids', '
fish_larvae_very_thin_body', 'fish_larvae_thin_body', 'tornaria_acorn_worm_larvae', 'copepod_calanoid_
eggs', 'hydromedusae_sideview_big', 'appendicularian_fritillaridae', 'siphonophore_calycophoran_abylid
ae', 'fish_larvae_deep_body', 'jellies_tentacles'] 0
```

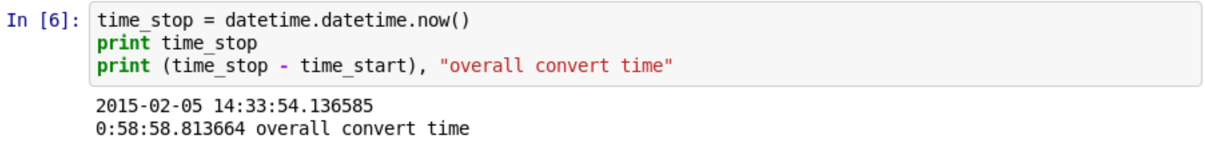


```
2 /home/efg/2015/Kaggle/Plankton/train/appendicularian_slight_curve [] 483
3 /home/efg/2015/Kaggle/Plankton/train/chaetognath_other [] 1934
4 /home/efg/2015/Kaggle/Plankton/train/hydromedusae_typeE [] 14
5 /home/efg/2015/Kaggle/Plankton/train/copepod_calanoid_large_side_antennatucked [] 106
6 /home/efg/2015/Kaggle/Plankton/train/copepod_calanoid_flatheads [] 178
7 /home/efg/2015/Kaggle/Plankton/train/shrimp-like_other [] 52
8 /home/efg/2015/Kaggle/Plankton/train/protist_dark_center [] 108
9 /home/efg/2015/Kaggle/Plankton/train/decapods [] 55
10 /home/efg/2015/Kaggle/Plankton/train/echinoderm_larva_pluteus_urchin [] 88
11 /home/efg/2015/Kaggle/Plankton/train/heteropod [] 10
12 /home/efg/2015/Kaggle/Plankton/train/acantharia_protist_big_center [] 13
13 /home/efg/2015/Kaggle/Plankton/train/hydromedusae_typeD_bell_and_tentacles [] 56
14 /home/efg/2015/Kaggle/Plankton/train/ctenophore_lobate [] 38
15 /home/efg/2015/Kaggle/Plankton/train/unknown_unclassified [] 425
16 /home/efg/2015/Kaggle/Plankton/train/protist_other [] 1172
17 /home/efg/2015/Kaggle/Plankton/train/fish_larvae_medium_body [] 85
18 /home/efg/2015/Kaggle/Plankton/train/copepod_calanoid_large [] 286
19 /home/efg/2015/Kaggle/Plankton/train/copepod_calanoid_octomoms [] 49
20 /home/efg/2015/Kaggle/Plankton/train/hydromedusae shapeB [] 150
```

. . .



```
In [5]: for i in range(len(DIRWALK)):
            print i, DIRWALK[i][0], len(DIRWALK[i][2])
            for j in range(len(DIRWALK[i][2])):
                filename = DIRWALK[i][2][j]
                if j > 0 and j % 2500 == 0:
                    print j, filename
                basename = os.path.join(DIRWALK[i][0], filename.rsplit(".",2)[0])
                # print j, filename
                subprocess.check_output(["convert", basename+".jpg", basename+".tif"])
                # Use print for debug
                # print subprocess.check_output(["convert", basename+".jpg", basename+".tif"])
                os.remove(basename+".jpg")
```



```
In [6]: time_stop = datetime.datetime.now()
        print time_stop
        print (time_stop - time_start), "overall convert time"

        2015-02-05 14:33:54.136585
        0:58:58.813664 overall convert time
```

[295 efg localhost 2015-02-05 19:46:50 /home/efg]
wndchrm

```
wndchrm 1.60.  Laboratory of Genetics/NIA/NIH
usage:
======
wndchrm [ train | test | classify | check ] [-mtslcdowfrijnpqvNSBACDTh] [<dataset>|<train set>] [<test set>|<feature
file>] [<report_file>]
  <dataset> is a <root directory>, <feature file>, <file of filenames>, <image directory> or <image filename>
  <root directory> is a directory of sub-directories containing class images with one class per sub-directory.
      The sub-directory names will be used as the class labels. Currently supported file formats: TIFF, PPM.
  <feature file> is the file generated by the train command containing all computed image features (should end in
.fit).
       This filename is a required parameter for storing the output of 'train'
  <file of filenames> is a text file listing <image filename>s and corresponding class labels
      separated by a <TAB> character (a tab delimited file, or .tsv). Lines beginning with '#' are ignored
  <image directory> is a directory of image files. The class cannot be specified so these can only be used as a <test
set>.
  <image filename> is the full path to an image file. The classes cannot be specified so these can only be used as a
<test set>.
  <train set> is anything that qualifies as a <dataset>, but must contain at least two (2) defined classes.
      An <image filename> or <image directory> cannot define classes.
  <test set> is anything that qualifies as a <dataset>.  The <train set> will be used to classify the <test set>.
      This parameter is required for 'classify' and is optional for 'test' (when doing internal tests of the <train
set>
  <report_file> is a report of the test/classify results in html format (must end in .htm or .html).

Image sampling options (require re-computing features):
========================================================
m - Allow running multiple instances of this program concurrently, save (and re-use) pre-calculated .sig files.
    This will save and re-use .sig files, making this option useful for single instances/processors as well
R - Add rotations to training images (0,90,180,270 degrees).
t[#][^]C[xR] - split the image into CxC or CxR tiles if R is specified. The default is 1.
    If '#' is specified, each tile location is used as a separate dataset (for testing only!).
      - If both '#' and '^' are specified only the closest tile is used.
    If only C is specified (e.g. -t2), tiling will be CxC (e.g. 2 columns by 2 rows).
      - If both C and R are specified (e.g. -t2x3), tiling will be CxR (e.g. 2 columns by 3 rows).
dN - Downsample the images (N percents, where N is 1 to 100)
Sx[:y] - normalize the images such that the mean is set to x and (optinally) the stddev is set to y.
Bx,y,w,h - compute features only from the (x,y,w,h) block of the image.

Image Feature options:
======================
```

WndCharm Notes

```
l - Use a large image feature set.
c - Compute color features.
o - force overwriting pre-computed .sig files.
O - if there are pre-computed .sig files accompanying images that have the old-style naming pattern,
    skip the check to see that they were calculated with the same wndchrm parameters as the current experiment.

Feature reduction options:
==========================
fN[:M] - maximum number of features out of the dataset (0,1) . The default is 0.15.
v[r|w|+|-][path] - read/write/add/subtract the feature weights from a file.
A - assess the contribution of each group of image features independently.

Classifier options:
===================
w - Classify with wnn instead of wnd.
qN - the number of first closest classes among which the presence of the right class is considered a match.
r[#]N - Fraction of images/samples to be used for training (0,1). The default is 0.75 of
        the smallest class. if '#' is specified, force unbalanced training
i[#]N - Set a maximal number of training images (for each class). If the '#' is specified then
        the class is ignored if it doesn't have at least N samples.
jN - Set a maximal number of test images (for each class).
nN - Number of repeated random splits. The default is 1.
Nx - set the maximum number of classes (use only the first x classes).

Output options:
===============
s - silent mode. Optionally followed by a verbosity level (higher = more verbose)
p[+][k][#][path] - Report options.
   'path' is an optional path to a PHYLIP installation root directory for generating dendrograms.
   The optinal '+' creates a 'tsv' directory and exports report data into tsv files.
   'k' is an optional digit (1..3) of the specific phylip algorithm to be used.
   '#' generates a similarity map of the test images
P[N] - pair-wise distance algorithms for comparing classes
   The class probability matrix is the average of marginal probabilities for the images in each class
   The similarity matrix is the class probability matrix, where each row is normalized to make the class identity
column equal to 1.0
   The dis-similarity (i.e. 1.0 - similarity) between two classes can be interpreted as a "morphological distance".
   There are two entries in the similarity matrix for each comparison: Class 1 classified as Class 2, and Class 2
classified as Class 1.
   N = 1: Use the maximum of the two dis-similarities.
   N = 2: Use the average of the two dis-similarities.
   N = 3: Use the top triangle only (i.e. Class 1 classified as Class 2)
   N = 4: Use the bottom triangle only (i.e. Class 2 classified as Class 1)
```

WndCharm Notes

```
   N = 5: Use the class probability matrix as a set of coordinates for each class centroid in a "Marginal Probability
Space". Use Euclidean distance.
   The default method is 5. Method 2 was described in ref [1], and method 5 was described in ref [2].
D[path] - feature file name (.fit file) to save the <dataset> or <train set>.
T[path] - feature file name (.fit file) to save the <test set>.
h - show this note.

Examples:
=========
train:
  wndchrm train /path/to/dataset/ dataset.fit
  wndchrm train -mcl /path/to/dataset/ testset.fit
test:
  wndchrm test -f0.1 dataset.fit
  wndchrm test -f0.1 -r0.9 -n5 dataset.fit testset.fit
  wndchrm test -f0.2 -i50 -j20 -n5 -p/path/to/phylip3.65 dataset.fit testset.fit report.html
  N.B.: By default, the -r or -i parameters will be used to make a balanced training set (equal number of images per
class).
       -r#N can be used to override this default, so that the N fraction of each class will be used for training.
       If a <test set> is specified, it will be used as the test set for each 'split', but training images will
       still be randomly chosen from <train set>)
classify:
   wndchrm classify dataset.fit /path/to/image.tiff
   wndchrm classify -f0.2 -cl /path/to/root/dir /path/to/image/directory/
   wndchrm classify -f0.2 -cl -Ttestset.fit dataset.fit /path/to/image/file_of_filenames.tsv
   N.B.: classify will use -r or -i to train with fewer than all of the images in <dataset>
       Unlike 'test', 'classify' will chose the training images in order rather than randomly.
       classify will ignore the -n parameter because the result will be the same for each run or split.
       The default -r for 'classify' is 1.0 rather than the 0.75 used in 'test'.
check: Report which features will need to be computed
   wndchrm check path/to/image.tiff

Additional help:
================
A detailed description can be found in: Shamir, L., Orlov, N., Eckley, D.M., Macura, T., Johnston, J., Goldberg, I.
  [1] "Wndchrm - an open source utility for biological image analysis", BMC Source Code for Biology and Medicine, 3:13,
2008.  http://ome.grc.nia.nih.gov/wnd-charm/BMC-wndchrm-utility.pdf

An application of pattern recognition for a quantitative biological assay based on morphology can be found in:
  [2] Johnston, J., Iser W. B., Chow, D. K., Goldberg, I. G., Wolkow, C. A. "Quantitative Image Analysis Reveals
  Distinct Structural Transitions during Aging in Caenorhabditis elegans Tissues", PLoS ONE, 3:7:e2821, 2008.

If you have questions or problems with this software, please visit our Google code page http://code.google.com/p/wnd-
charm/
```

WndCharm Notes

Using the Command Line Utility

This paper describes the command-line parameters:

http://ome.grc.nia.nih.gov/wnd-charm/BMC-wndchrm-utility.pdf, p. 5/13

Train

In order to train an image classifier, the first required task is computing image content descriptors for all images in the dataset. These numeric values describe the image content in a fashion that can later be processed by pattern recognition methods. This step is performed by using a simple command line described below:

```
wndchrm train [options] images feature_file
```

where ***feature_file*** is the resulting output file of the image feature values, ***images*** is a path to the top folder where the images of the dataset are stored, and [*options*] are optional switches that can be specified by the user. The top folder should consist of several sub-folders such that each subfolder contains images of a different class.

The single output file ***feature_file*** contains features for all classes in the dataset, so that there are no separate files for the different classes. Therefore, if a new class is added to the dataset, a new file needs to be created using the same command line. To avoid re-computing classes that have already been computed, the user is advised to use the "-m" switch that will be described later in this section.

Test

Once all image content descriptors are computed, the dataset can be tested for classification accuracy. This can be done by using the following command line:

```
wndchrm test [options] feature_file [report_file]
```

where ***feature_file*** is the output file of the train task, and ***report_file*** is an optional html file providing detailed information regarding the performance of the classifier. This instruction automatically splits the images of each class into training and test images, and the effectiveness of the classifier is determined by the percentage of test images that are classified correctly using the training images. The test images are classified by computing the Fisher scores and assigning the image features with weights. The output of this command is a confusion matrix, a similarity matrix, and the accuracy of the classifier (the percentage of test images that were classified correctly).

When testing an image classifier, the user can determine the number of images that are used for testing and the number of images used for training. By default, 75% of the images of each class are used for the training, and the remaining 25% are used for testing. The user can change this ratio by using the "-r" option. For example, "-r0.4" allocates 40% of the images for testing, and 60% for training. The allocation of the images to training and test sets is performed in a random fashion. Users can repeat the test with several different random splits in a single command by specifying the "-n" option, followed by the requested number of splits.

wndchrm also allows the user to set the number of training images per class. This can be done using the "-i" option, followed by the requested number of training images per class. The remaining images are used for testing, unless the "-j" option is used in a similar fashion to set the number of test images per class. It should be noted that "-i" and "-

j" options override the "-r" value. If only one of these options is specified, and "-r" is also used, the number of training or test images per class (the one that is not determined by "-i" or "-j") will be determined by the "-r" value. **We suggest using a fixed number of training images per class ("-i") when generating similarity matrices.**

Users can also use different feature files (generated by using *wndchrm*'s "train" command) for testing and training, so that instead of splitting a single dataset into training and test images, one dataset is used entirely for training while a second dataset is used for testing. This can be done by **simply specifying two full paths to image feature files. If two files are specified, the first will be used for training and the second for testing.**

Classify

After a classifier is trained and tested, an image can be classified using the command line:

```
wndchrm classify feature_file image
```

where ***image*** can be a full path to the image being classified, or a folder that contains multiple *images*. If image points to a specific image file, the output of this instruction is the predicted class in which the image belongs, as well as a vector of similarity values to each of the classes in the dataset. If *image* is a path to a folder, *wndchrm* classifies and prints the predicted class and similarity vector for each image in that folder, followed by a brief summary that specifies the number of images that were classified to each class and the average similarity vector.

Changing the Number of Image Features

Since *wndchrm* is a multi-purpose tool designed to handle many different image datasets, it uses very many different image features. However, for a given dataset, not all image features are assumed to be equally informative, and some of these features are expected to represent noise. By default, only the 0.15 images features with the highest Fisher scores are used. Users who wish to change this setting can specify the "-f" option in the command line, followed by the requested portion of the image features to be used. Changing this value can affect the performance of the image analysis since in some datasets more image features may be informative, so that using more features can contribute to the discrimination between the classes. On the other hand, in other datasets only few of the image features provide discriminative information, and using

the non-informative features can add confusion and degrade the efficacy of the analysis. Since image features are weighed by their informativeness, the effect of noisy features is expected to be lower than the effect of more informative features. However, if very many non-informative image features are used, their large number can be weighed against their low Fisher scores, leading to an undesirable degradation of the performance. **Therefore, the threshold for non-informative features needs to be determined operationally for each type of data, and the 0.15 threshold is only a starting point.**

Image Tiling

In some cases it may be useful to divide large images of tissues or cells into several equal-sized tiles. For example, it has been demonstrated that when each image captures very many cells, dividing the image into tiles can in some cases provide better analysis than applying a first step of global-thresholding cell segmentation [22]. Another advantage is that using more tiles can improve the effectiveness of the Fisher scores assigned to the image features, which are expected to improve as the size of the dataset gets larger. Using *wndchrm*, this can be done by specifying the "-t" option followed by the square root of the desired number of tiles. For instance, "-t3" divides each image into 3 × 3 tiles.

If segmentation of the subjects (e.g., cell segmentation, bone segmentation, etc) is required, the user has to apply a first step of segmentation using a designated utility. The output of the utility (the segmented subjects) can be used as input for *wndchrm*, rather than the original images

Phylip Installation

http://evolution.genetics.washington.edu/phylip.html

make install

```
[302 efg localhost 2015-02-05 18:59:30 /home/efg/Software/Phylip/phylip-3.696]
ll
total 20
drwxr-xr-x. 3 efg efg 4096 Feb  5 18:45 doc
drwxr-xr-x. 2 efg efg 4096 Feb  5 18:48 exe
-rw-r--r--. 1 efg efg 7593 Sep 21 12:19 phylip.html
drwxr-xr-x. 6 efg efg 4096 Feb  5 18:48 src
```

Train

```
[298 efg localhost 2015-02-05 19:02:46 /home/efg/2015/Kaggle/Plankton]
ll
total 7640
drwxrwxrwx.   2 efg efg 7819264 Feb  5 14:33 test
drwxrwxrwx. 123 efg efg    4096 Dec 30 12:19 train
```

To use four processors:

```
cd /home/efg/2015/Kaggle/Plankton
```

```
wndchrm train -ml /home/efg/2015/Kaggle/Plankton/train Plankton-01.fit
wndchrm train -ml /home/efg/2015/Kaggle/Plankton/train Plankton-01.fit
wndchrm train -ml /home/efg/2015/Kaggle/Plankton/train Plankton-01.fit
wndchrm train -ml /home/efg/2015/Kaggle/Plankton/train Plankton-01.fit
```

```
-m:  Allow running multiple instances of this program concurrently.
-l:  Use large image feature set.
```

Test

```
wndchrm test -f0.25 -r0.75 -n5 -p/home/efg/Software/Phylip/phylip-3.696 Plankton-01.fit Plankton-01.html
```

```
[346 efg localhost 2015-02-06 06:53:20 /home/efg/2015/Kaggle/Plankton]
```

```
wndchrm test -f0.2 -n5 -p/home/efg/Software/Phylip/phylip-3.696 Plankton-01.fit Plankton-01.html
```

```
wndchrm test -f0.2 -i50 -n5 -p/home/efg/Software/Phylip/phylip-3.696 Plankton-01.fit Plankton-01b.html
```

```
[358 efg localhost 2015-02-06 08:06:42 /home/efg/2015/Kaggle/Plankton]
```

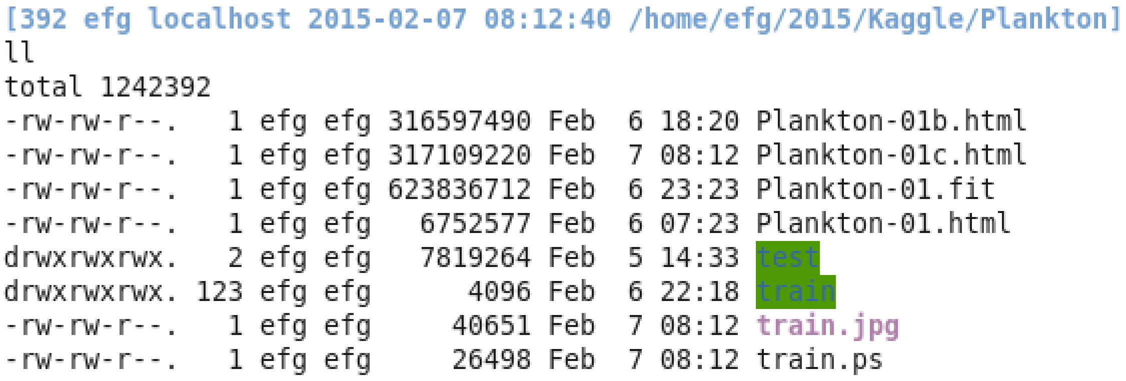
```
wndchrm test -l -f0.2 -r#1 -n5 -p/home/efg/Software/Phylip/phylip-3.696
/home/efg/2015/Kaggle/Plankton/train  /home/efg/2015/Kaggle/Plankton/train Plankton-
01b.html
```

Force Train and Test on all images regardless of class size:

```
wndchrm test -l -f0.2 -r#1 -n5 -p/home/efg/Software/Phylip/phylip-3.696/ /home/efg/2015/Kaggle/Plankton/train/
/home/efg/2015/Kaggle/Plankton/train/ Plankton-01c.html
```

```
[391 efg localhost 2015-02-06 23:32:31 /home/efg/2015/Kaggle/Plankton]
wndchrm test -l -f0.2 -r#1 -n5 -p/home/efg/Software/Phylip/phylip-3.696/ /home/efg/2015/Kaggle/Plankton/train/
/home/efg/2015/Kaggle/Plankton/train/ Plankton-01c.html
Processing training set '/home/efg/2015/Kaggle/Plankton/train/'.
. . .

118. g_center
```

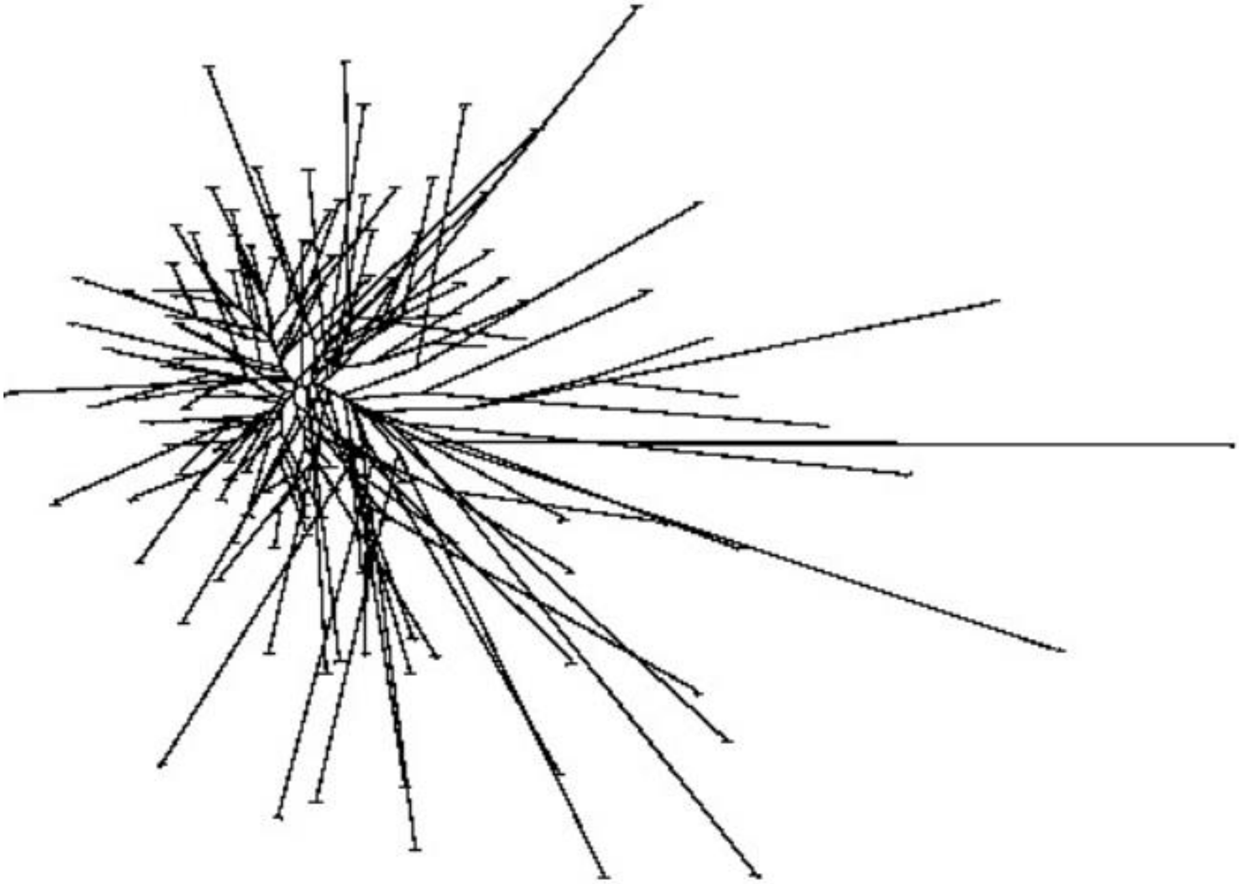
```
 119. ucalanus
 120. _copilia
 121. ew_small

Output written to file "outfile"

Tree also written onto file "outtree"

Done.


DRAWTREE from PHYLIP version 3.696
drawtree: can't find input tree file "intree"
Please enter a new file name> Reading tree ...
Tree has been read.
Loading the font ...
drawtree: can't find font file "fontfile"
Please enter a new file name> Font loaded.

Unrooted tree plotting program version 3.696

Here are the settings:

 0  Screen type (IBM PC, ANSI)?  ANSI
 P       Final plotting device:  Postscript printer
 (Preview not available)
 B          Use branch lengths:  Yes
 L             Angle of labels:  branch points to Middle of label
 R            Rotation of tree:  90.0
 I     Iterate to improve tree:  Equal-Daylight algorithm
 D  Try to avoid label overlap?  No
 S      Scale of branch length:  Automatically rescaled
 C   Relative character height:  0.3333
 F                        Font:  Times-Roman
 M          Horizontal margins:  1.65 cm
 M            Vertical margins:  2.16 cm
 #           Page size submenu:  one page per tree

 Y to accept these or type the letter for one to change


Unrooted tree plotting program version 3.696

Here are the settings:

 0  Screen type (IBM PC, ANSI)?  ANSI
 P       Final plotting device:  Postscript printer
 (Preview not available)
 B          Use branch lengths:  Yes
 L             Angle of labels:  branch points to Middle of label
 R            Rotation of tree:  90.0
 I     Iterate to improve tree:  n-Body algorithm
 D  Try to avoid label overlap?  No
 S      Scale of branch length:  Automatically rescaled
 C   Relative character height:  0.3333
 F                        Font:  Times-Roman
 M          Horizontal margins:  1.65 cm
 M            Vertical margins:  2.16 cm
 #           Page size submenu:  one page per tree

 Y to accept these or type the letter for one to change



Unrooted tree plotting program version 3.696

Here are the settings:

 0  Screen type (IBM PC, ANSI)?  ANSI
 P       Final plotting device:  Postscript printer
 (Preview not available)
 B          Use branch lengths:  Yes
 L             Angle of labels:  branch points to Middle of label
 R            Rotation of tree:  90.0
 I     Iterate to improve tree:  n-Body algorithm
 D  Try to avoid label overlap?  No
 S       Scale of branch length:  Automatically rescaled
 C   Relative character height:  0.3333
 F                         Font:  Times-Roman
 M          Horizontal margins:  1.65 cm
 M            Vertical margins:  2.16 cm
 #           Page size submenu:  one page per tree

 Y to accept these or type the letter for one to change



Unrooted tree plotting program version 3.696

Here are the settings:

 0  Screen type (IBM PC, ANSI)?  ANSI
 P       Final plotting device:  Postscript printer
 (Preview not available)
 B          Use branch lengths:  Yes
 L             Angle of labels:  branch points to Middle of label
 R            Rotation of tree:  90.0
 I     Iterate to improve tree:  n-Body algorithm
 D  Try to avoid label overlap?  No
 S       Scale of branch length:  Automatically rescaled
 C   Relative character height:  0.3333
 F                         Font:  Times-Roman
 M          Horizontal margins:  1.65 cm
 M            Vertical margins:  2.16 cm
 #           Page size submenu:  one page per tree

 Y to accept these or type the letter for one to change

Writing plot file ...

Plot written to file "plotfile"

Done.
```



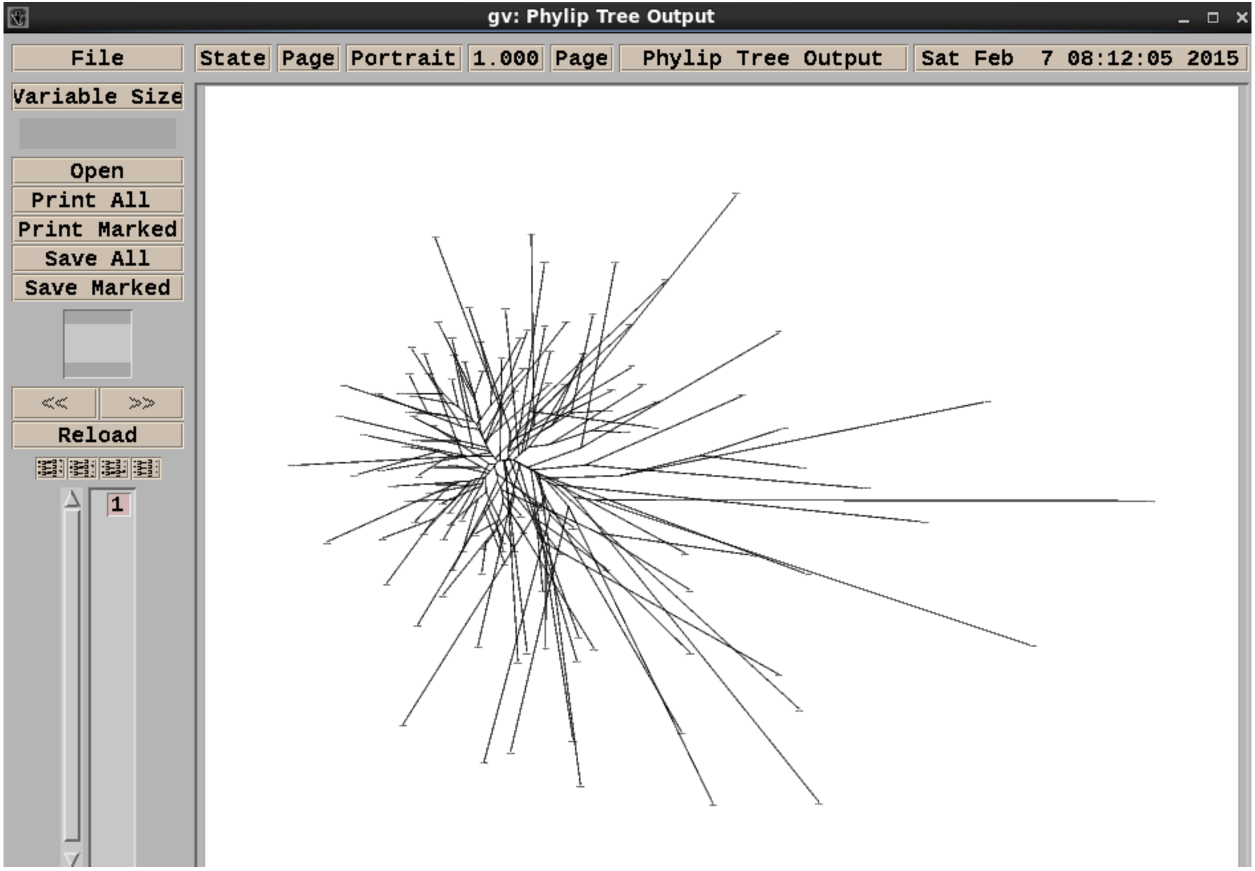
```
[392 efg localhost 2015-02-07 08:12:40 /home/efg/2015/Kaggle/Plankton]
ll
total 1242392
-rw-rw-r--.   1 efg efg 316597490 Feb  6 18:20 Plankton-01b.html
-rw-rw-r--.   1 efg efg 317109220 Feb  7 08:12 Plankton-01c.html
-rw-rw-r--.   1 efg efg 623836712 Feb  6 23:23 Plankton-01.fit
-rw-rw-r--.   1 efg efg   6752577 Feb  6 07:23 Plankton-01.html
drwxrwxrwx.   2 efg efg   7819264 Feb  5 14:33 test
drwxrwxrwx. 123 efg efg      4096 Feb  6 22:18 train
-rw-rw-r--.   1 efg efg     40651 Feb  7 08:12 train.jpg
-rw-rw-r--.   1 efg efg     26498 Feb  7 08:12 train.ps
```

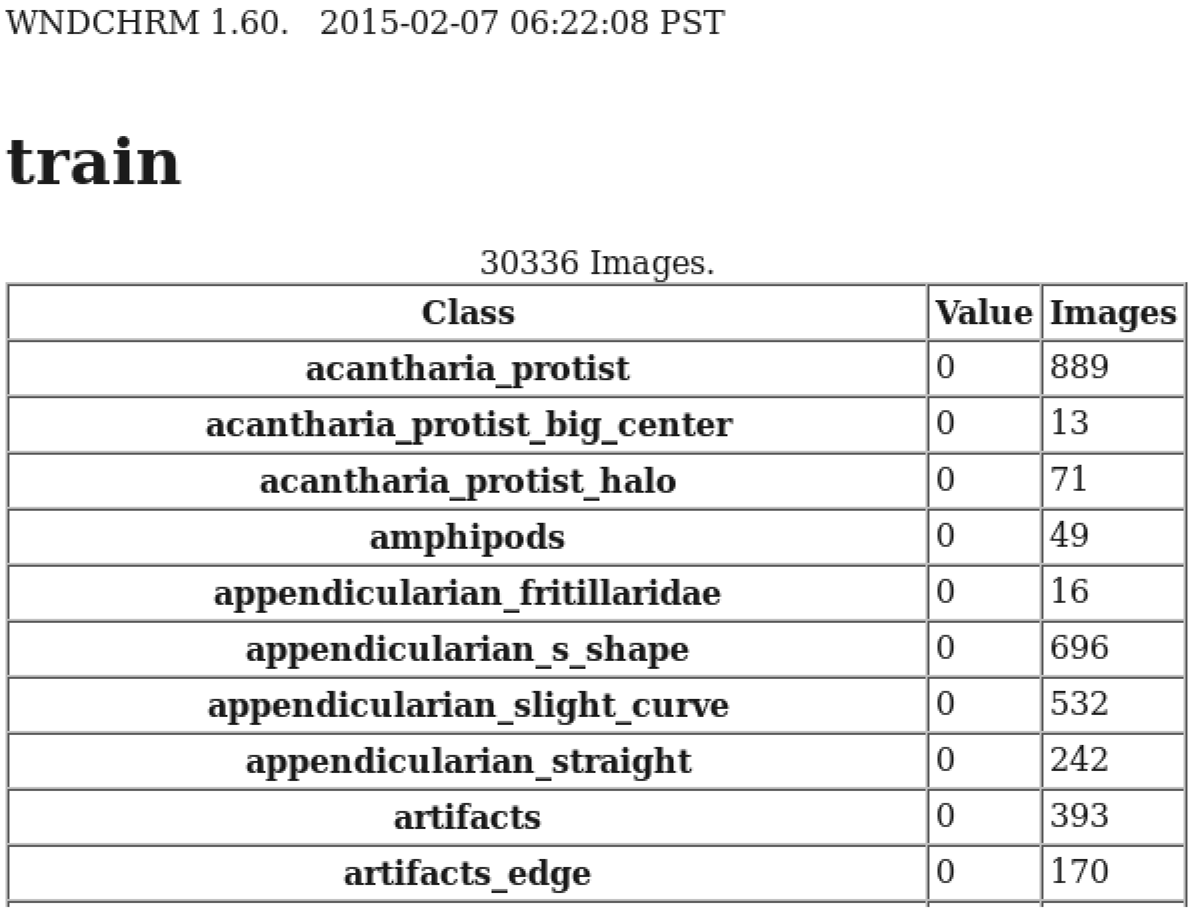
WndCharm Notes

`train.jpg`



```
train.ps

gv train.ps
```





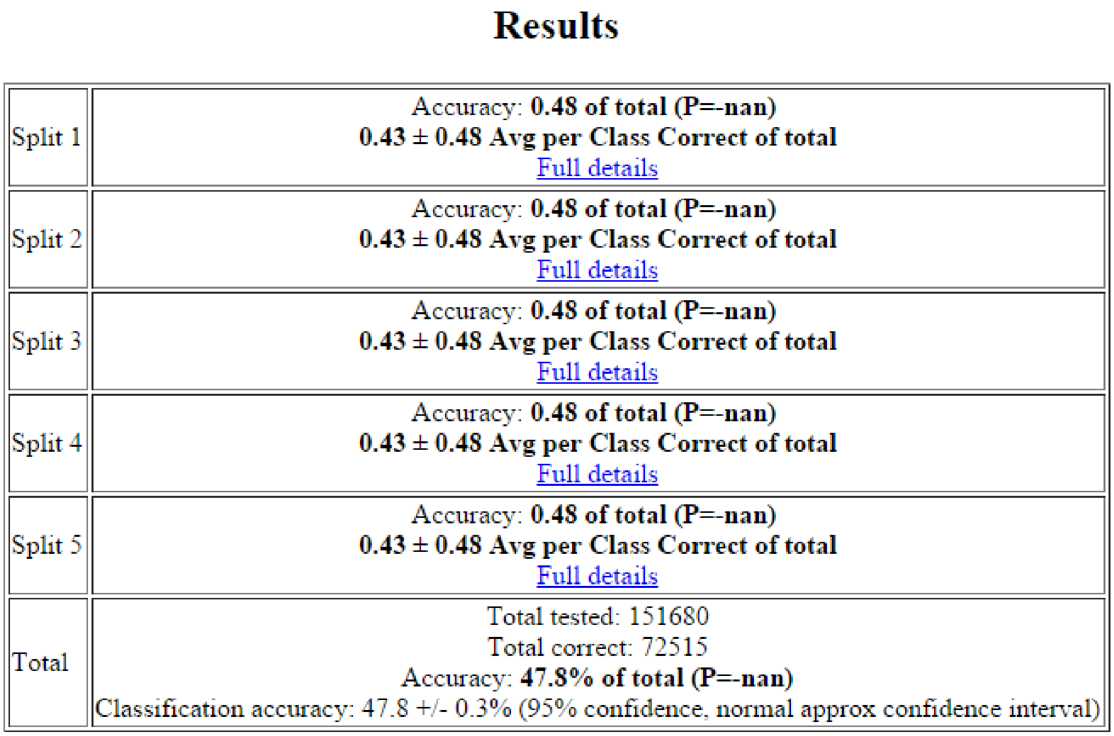
WNDCHRM 1.60. 2015-02-07 06:22:08 PST

train

30336 Images.

Class	Value	Images
acantharia_protist	0	889
acantharia_protist_big_center	0	13
acantharia_protist_halo	0	71
amphipods	0	49
appendicularian_fritillaridae	0	16
appendicularian_s_shape	0	696
appendicularian_slight_curve	0	532
appendicularian_straight	0	242
artifacts	0	393
artifacts_edge	0	170

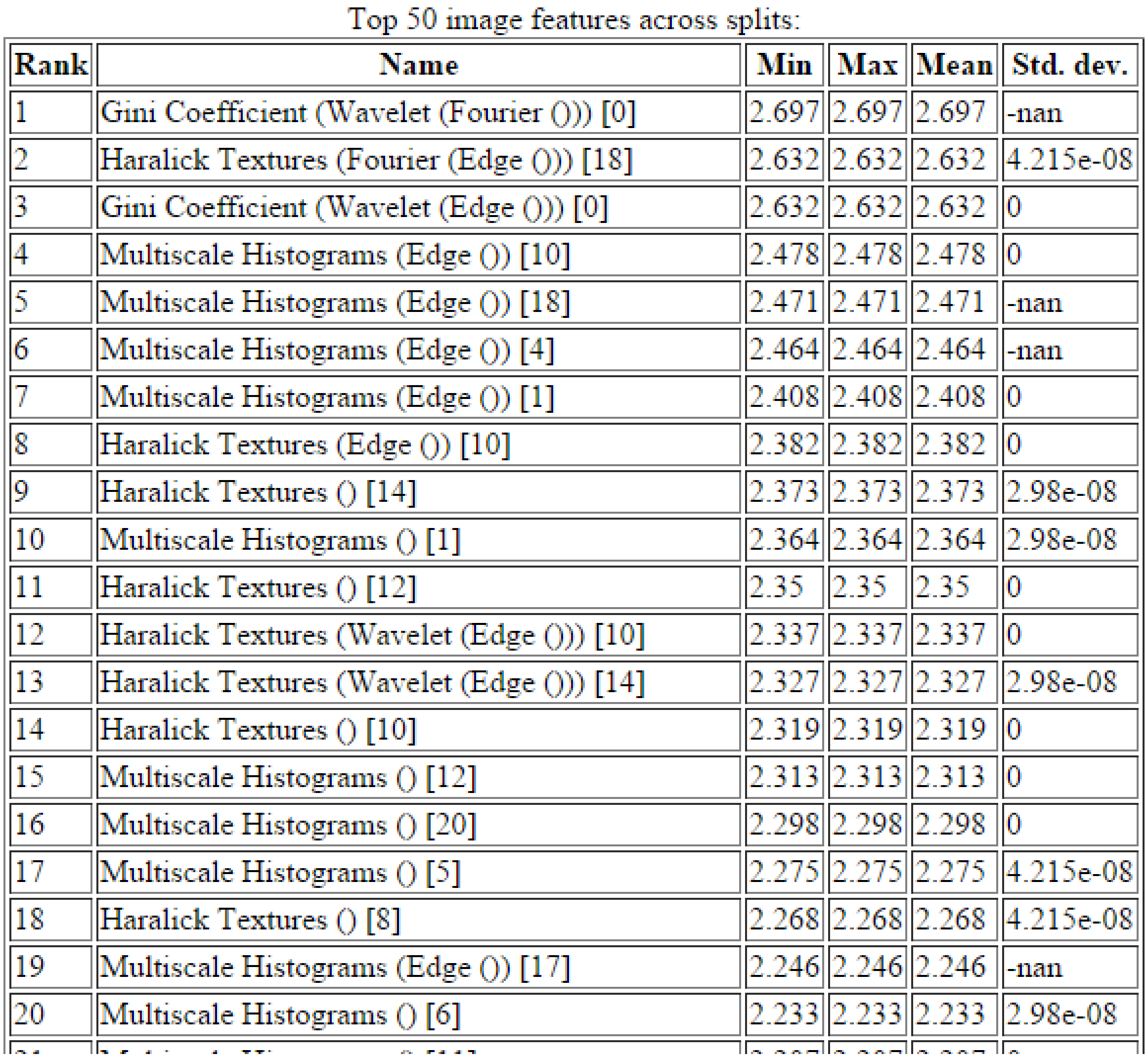
```
Firefox freezes on CentOS with 300+MB HTML file.  Transfer to Windows and view in Chrome.
```



Results

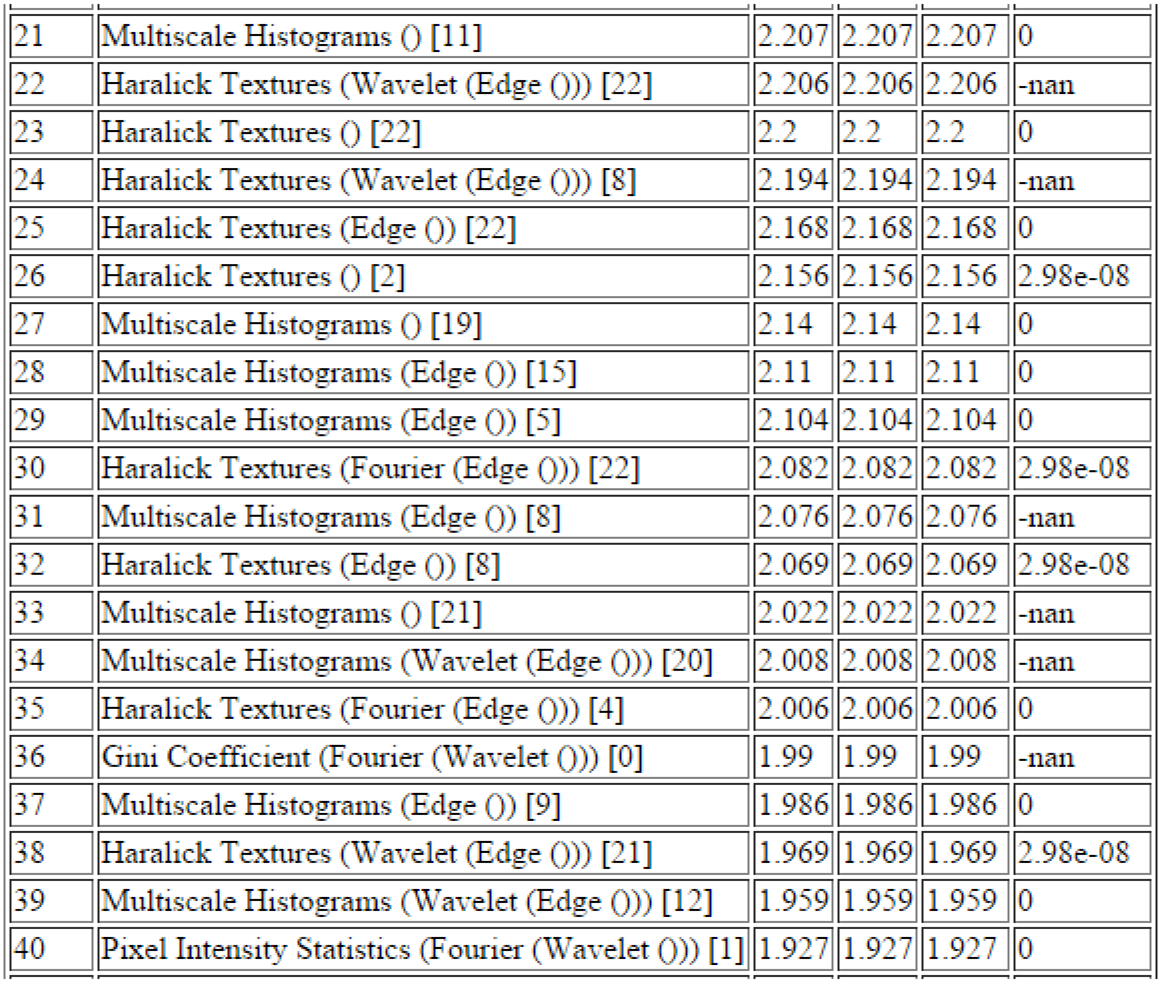
Split 1	Accuracy: **0.48 of total (P=-nan)** **0.43 ± 0.48 Avg per Class Correct of total** Full details
Split 2	Accuracy: **0.48 of total (P=-nan)** **0.43 ± 0.48 Avg per Class Correct of total** Full details
Split 3	Accuracy: **0.48 of total (P=-nan)** **0.43 ± 0.48 Avg per Class Correct of total** Full details
Split 4	Accuracy: **0.48 of total (P=-nan)** **0.43 ± 0.48 Avg per Class Correct of total** Full details
Split 5	Accuracy: **0.48 of total (P=-nan)** **0.43 ± 0.48 Avg per Class Correct of total** Full details
Total	Total tested: 151680 Total correct: 72515 Accuracy: **47.8% of total (P=-nan)** Classification accuracy: 47.8 +/- 0.3% (95% confidence, normal approx confidence interval)

WndCharm Notes

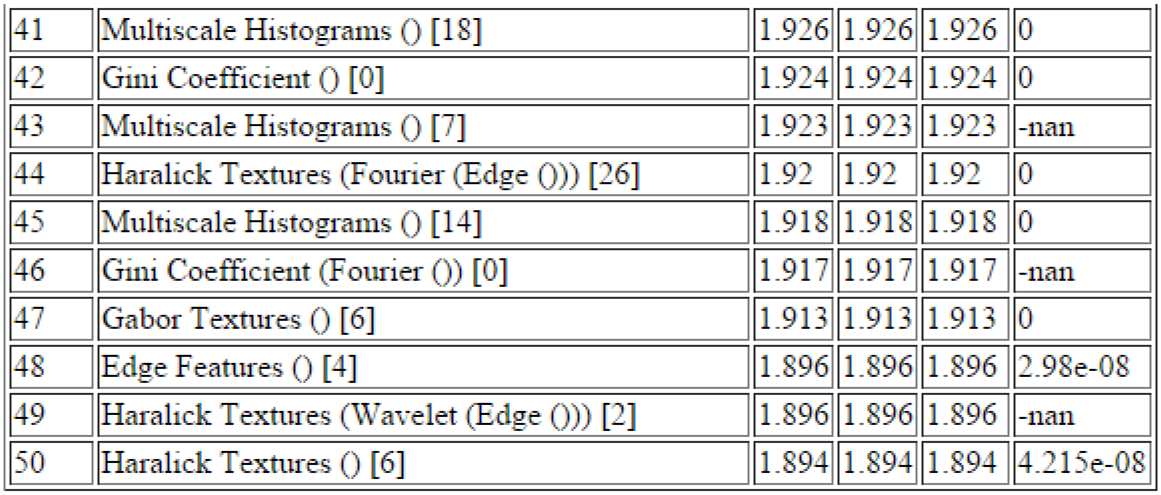


Top 50 image features across splits:

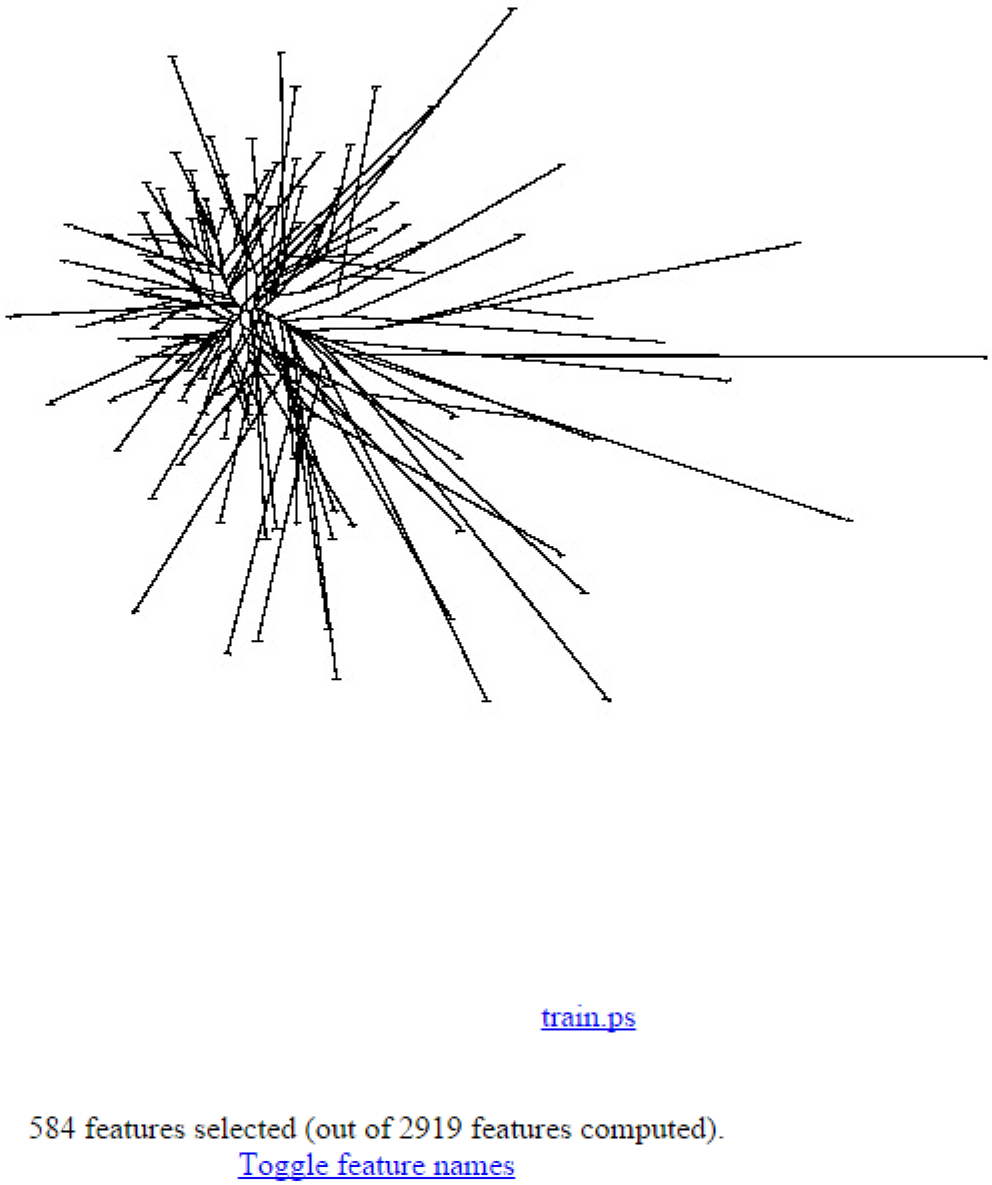
Rank	Name	Min	Max	Mean	Std. dev.
1	Gini Coefficient (Wavelet (Fourier ())) [0]	2.697	2.697	2.697	-nan
2	Haralick Textures (Fourier (Edge ())) [18]	2.632	2.632	2.632	4.215e-08
3	Gini Coefficient (Wavelet (Edge ())) [0]	2.632	2.632	2.632	0
4	Multiscale Histograms (Edge ()) [10]	2.478	2.478	2.478	0
5	Multiscale Histograms (Edge ()) [18]	2.471	2.471	2.471	-nan
6	Multiscale Histograms (Edge ()) [4]	2.464	2.464	2.464	-nan
7	Multiscale Histograms (Edge ()) [1]	2.408	2.408	2.408	0
8	Haralick Textures (Edge ()) [10]	2.382	2.382	2.382	0
9	Haralick Textures () [14]	2.373	2.373	2.373	2.98e-08
10	Multiscale Histograms () [1]	2.364	2.364	2.364	2.98e-08
11	Haralick Textures () [12]	2.35	2.35	2.35	0
12	Haralick Textures (Wavelet (Edge ())) [10]	2.337	2.337	2.337	0
13	Haralick Textures (Wavelet (Edge ())) [14]	2.327	2.327	2.327	2.98e-08
14	Haralick Textures () [10]	2.319	2.319	2.319	0
15	Multiscale Histograms () [12]	2.313	2.313	2.313	0
16	Multiscale Histograms () [20]	2.298	2.298	2.298	0
17	Multiscale Histograms () [5]	2.275	2.275	2.275	4.215e-08
18	Haralick Textures () [8]	2.268	2.268	2.268	4.215e-08
19	Multiscale Histograms (Edge ()) [17]	2.246	2.246	2.246	-nan
20	Multiscale Histograms () [6]	2.233	2.233	2.233	2.98e-08
21	Multiscale Histograms () [11]	2.207	2.207	2.207	0



21	Multiscale Histograms () [11]	2.207	2.207	2.207	0
22	Haralick Textures (Wavelet (Edge ())) [22]	2.206	2.206	2.206	-nan
23	Haralick Textures () [22]	2.2	2.2	2.2	0
24	Haralick Textures (Wavelet (Edge ())) [8]	2.194	2.194	2.194	-nan
25	Haralick Textures (Edge ()) [22]	2.168	2.168	2.168	0
26	Haralick Textures () [2]	2.156	2.156	2.156	2.98e-08
27	Multiscale Histograms () [19]	2.14	2.14	2.14	0
28	Multiscale Histograms (Edge ()) [15]	2.11	2.11	2.11	0
29	Multiscale Histograms (Edge ()) [5]	2.104	2.104	2.104	0
30	Haralick Textures (Fourier (Edge ())) [22]	2.082	2.082	2.082	2.98e-08
31	Multiscale Histograms (Edge ()) [8]	2.076	2.076	2.076	-nan
32	Haralick Textures (Edge ()) [8]	2.069	2.069	2.069	2.98e-08
33	Multiscale Histograms () [21]	2.022	2.022	2.022	-nan
34	Multiscale Histograms (Wavelet (Edge ())) [20]	2.008	2.008	2.008	-nan
35	Haralick Textures (Fourier (Edge ())) [4]	2.006	2.006	2.006	0
36	Gini Coefficient (Fourier (Wavelet ())) [0]	1.99	1.99	1.99	-nan
37	Multiscale Histograms (Edge ()) [9]	1.986	1.986	1.986	0
38	Haralick Textures (Wavelet (Edge ())) [21]	1.969	1.969	1.969	2.98e-08
39	Multiscale Histograms (Wavelet (Edge ())) [12]	1.959	1.959	1.959	0
40	Pixel Intensity Statistics (Fourier (Wavelet ())) [1]	1.927	1.927	1.927	0



41	Multiscale Histograms () [18]	1.926	1.926	1.926	0
42	Gini Coefficient () [0]	1.924	1.924	1.924	0
43	Multiscale Histograms () [7]	1.923	1.923	1.923	-nan
44	Haralick Textures (Fourier (Edge ())) [26]	1.92	1.92	1.92	0
45	Multiscale Histograms () [14]	1.918	1.918	1.918	0
46	Gini Coefficient (Fourier ()) [0]	1.917	1.917	1.917	-nan
47	Gabor Textures () [6]	1.913	1.913	1.913	0
48	Edge Features () [4]	1.896	1.896	1.896	2.98e-08
49	Haralick Textures (Wavelet (Edge ())) [2]	1.896	1.896	1.896	-nan
50	Haralick Textures () [6]	1.894	1.894	1.894	4.215e-08



train.ps

584 features selected (out of 2919 features computed).
Toggle feature names