SIPPER Plankton Classification System

Installation and User Manual

Kurt Kramer

2009-December- 28th

Department of Computer Science and Engineering

University of South Florida

[1 Introduction 4](#_Toc326836494)

[2 Installation. 4](#_Toc326836495)

[2.1 Server Installation. 4](#_Toc326836496)

[2.1.1 Install MySQL. 4](#_Toc326836497)

[2.1.2 Installation procedure using MySQL Workbench. 6](#_Toc326836498)

[2.2 Client Installation. 8](#_Toc326836499)

[2.2.1 Set Up PicesHomeDirectory Environment Variable. 8](#_Toc326836500)

[2.2.2 Run Setup Program. 11](#_Toc326836501)

[2.2.3 Copy over Needed Files. 13](#_Toc326836502)

[2.2.4 Edit Configuration files, such as MySQ.cfg configuration file. 14](#_Toc326836503)

[3 Backing Up Database. 16](#_Toc326836504)

[4 Restoring the Database 19](#_Toc326836505)

[5 Operations Software. 20](#_Toc326836506)

[5.1 SIPPER Interface. 20](#_Toc326836507)

[5.2 SIPPER Disk Manager. 20](#_Toc326836508)

[6 PICES Applications, Classification and Analysis Software. 20](#_Toc326836509)

[6.1 PICES Terminology. 21](#_Toc326836510)

[6.2 SIPPER File Viewer 22](#_Toc326836511)

[6.3 PICES Commander. 24](#_Toc326836512)

[6.3.1 Introduction. 24](#_Toc326836513)

[6.3.2 File Functions. 27](#_Toc326836514)

[6.3.3 Maintenance Functions. 27](#_Toc326836515)

[6.3.4 Image Classification. 32](#_Toc326836516)

[6.3.5 Create a Training Model. 33](#_Toc326836517)

[6.3.6 Group Assignments. 33](#_Toc326836518)

[6.4 Image Extraction and Classification 36](#_Toc326836519)

[6.5 Grading Function (Command Line) 38](#_Toc326836520)

[6.6 Grading Function (GUI) 39](#_Toc326836521)

[6.7 Image Classification (Command Line) 41](#_Toc326836522)

[6.8 Spatial Analysis 42](#_Toc326836523)

[6.9 Our Neighbors 42](#_Toc326836524)

[6.10 Delete Duplicate Images 43](#_Toc326836525)

[6.11 Cross Validation. 44](#_Toc326836526)

[6.12 Random Splits. 46](#_Toc326836527)

[6.13 Feature File Converter 47](#_Toc326836528)

[6.14 Feature File Stats 48](#_Toc326836529)

[6.15 Randomize Data 48](#_Toc326836530)

[6.16 Merge Feature Files 48](#_Toc326836531)

[6.17 Build Image Directory from a Feature File. 48](#_Toc326836532)

[6.18 SIPPER3 Simulator. 48](#_Toc326836533)

[7 Appendixes 49](#_Toc326836534)

[7.1 System Design. 49](#_Toc326836535)

[7.1.1 Data Flow in Sipper System 49](#_Toc326836536)

[7.2 Feature Descriptions. 53](#_Toc326836537)

[7.3 SIPPER3 Raw Data Layout 63](#_Toc326836538)

[7.4 Feature Data File Formats. 65](#_Toc326836539)

[7.5 Software Architecture. 65](#_Toc326836540)

[7.6 Database Description. 66](#_Toc326836541)

[7.6.1 Table Descriptions: 67](#_Toc326836542)

[7.6.2 Important Fields. 68](#_Toc326836543)

[7.6.3 Useful Scripts 68](#_Toc326836544)

[7.7 Future Work 68](#_Toc326836545)

[7.7.1 Training Model Maintenance Application. 68](#_Toc326836546)

[7.7.2 Support Vector Locator. 68](#_Toc326836547)

[7.7.3 Reasonable Training Image Locator. 69](#_Toc326836548)

[7.7.4 Full GUI Based Installation Program. 69](#_Toc326836549)

[7.8 Publications 70](#_Toc326836550)

[7.8.1 SIPPER Journal Papers 70](#_Toc326836551)

[7.8.2 SIPPER Conference Publications 70](#_Toc326836552)

[7.8.3 Other References 70](#_Toc326836553)

[Refernces 71](#_Toc326836554)

# Introduction

This document is meant to be the manual for SIPPER Software release 1.5 to be used as both a reference manual and a guide. The software in SIPPER can be broken down into two major groupings, Operations and Classification and Analysis. The operations part is what is required to operate SIPPER in the field; deploying SIPPER and the off loading of from the SIPPER drive. The analysis part is where discrete plankton images are extracted and classified into user defined classes other functions in this section include database management of the large number of images that SIPPER collect, reporting of classification results and tools that aid in the refinement in the image classifier.

# Installation.

Installation will require the assistance of your local technical staff. There are two types of installation that can be done, Server and Client. You will need to install only one Server installation and one or more client installations. The Server and Client installation can be on the same machine. The client installation is relatively simple require just basic windows experience such as how to use xcopy and setup environment variables. The Server installation will require someone with solid technical experience. You will first need to perform the Server Installation followed by all the client installations.

## Server Installation.

The server installation is requires the installation of a MySQL Database server, the creating of database tables and the installation of MySQL scripts. It can be installed on any kind of computer that is supported by MySQL, this includes Microsoft Windows and most Linux and Unix based systems plus many other O/S’s. See [www.mysql.com](http://www.mysql.com) for a list of supported platforms. You will need someone who is proficient in the O/S in questions and comfortable with installing SQL stored procedures.

### Install MySQL.

This is going to require someone who is very technically proficient. It is suggested that you get the aid of your local technical support. The server can be installed on any type of machine that MySQL server supports. You can also make use of any existing MySQL Installation to install the database on. You will also need to install the MySQL Client Tools or any other tool such as MySQL Workbench that will allow you to run MySQL scripts.

The latest version of MySQL can be downloaded from [www.mysql.com](http://www.mysql.com), the community server is free and is all that is required. Version 5.1 or later is required.

Requirements:

* + MySQL Server 5.1 or later.

MySQL Client GUI Tools or MySQL Workbench or any other tools that will allow you to run MySQL scripts.

* + At least 100 gig of free disk space. Ex: Our installation contains 24 million images that resulted from two dozen SIPPER deployments over 5 cruises uses approximately 70 gig of disk space on out MySQL server.
  + Suggest at least 4 gig of memory of machine that will run MySQL server.

These next Steps are a suggested guideline for your local technical support.

1. Download MySQL Server Community Server, ver 5.1 or later.
2. Install MySQL Server on appropriate machine, follow instructions provided by MySQL.
3. Suggested Configuration Parameters.

On the Install CD there is a MySQL configuration file y “\MySQL\my.ini” that is used on our installation. Do not use this file; but use it as guidance for configuring your own installation.

1. Create a DataBase where you will be creating the PICES Tables in. This can be any name but suggest “Pices”.
2. Create PICES Tables

Under the directory “\MySQL\Create Scripts\” there are the scripts for creating all the tables needed by PICES. (See Table 1).

Each row in Table 1 represents a table that needs to get created. You should create these tables in the same order as they appear in the table. Using a program like MSQL Query execute the create statement in each of the files listed. On some entries in Table 1 there will be some Insert statements that need to be executed.

1. Create PICES Scripts.

The source code to the stored procedures can be found in “\MySQL\StoredProcedures\”. Table 2 lists all the stored procedures that need to be created. The first column indicates the file name that contains the source code to the stored procedure while the second column contains the name of the stored procedure. Use the table as a check list to insure the installation of all the stored procedures. You can use a program like MySQL query to install the stored procedures. There is a short procedure just before Table 2 that describes installing the scripts using MySQL Workbench.

1. Create MySQL user and grant rights.

It is recommended that you create a specific MySQL user for working with PICES. This is not required but do not use the “root” user. Once you create this user you will need to grant this user access to the Stored procedures and tables.

1. Record the following fields that will be needed by the client installation.

|  |  |
| --- | --- |
| HostName | IP Address of machine running the MySQL Server. |
| UserName | MySQL User name that you created. |
| Password | Password to user. |
| DataBaseName | Database name you created in step 4 above. |

Table MySQL Create Scripts

|  |  |  |
| --- | --- | --- |
| **Source File name** | **Table Name** | **Other** |
| PicesDataBase.dbo.CtdSensors.SQL | CtdExternalSensors | Also execute the 4 Insert statements. |
| PicesDataBase.dbo.Instruments.SQL | Instruments | Also execute the 4 Insert statements. |
| PicesDataBase.dbo.Classes.SQL | Classes | Also execute the 2 Insert statements |
| PicesDataBase.dbo.Cruises.SQL | Cruises |  |
| PicesDataBase.dbo.Stations.SQL | Stations |  |
| PicesDataBase.dbo.Deployments.SQL | Deployments |  |
| PicesDataBase.dbo.SipperFiles.SQL | SipperFiles |  |
| PicesDataBase.dbo.Images.SQL | Images |  |
| PicesDataBase.dbo.ImagesFullSize.sql | ImagesFullSize |  |
| PicesDataBase.dbo.FeatureData.sql | FeatureData |  |
| PicesDataBase.dbo.InstrumentData.SQL | InstrumentData |  |
| PicesDataBase.dbo.ImageGroup.SQL | ImageGroup |  |
| PicesDataBase.dbo.ImageGroupEntries.SQL | ImageGroupEntries |  |

### Installation procedure using MySQL Workbench.

1) Under "SQL Development"

select "Open Connection to Start Querying"

select 'localhost'

2) Make pices\_new your default schema

do this by right mousing on "pices\_new" and select "Set defaul Schema"

3) for each Script file

Copy and paste each script file one at a time into scratch window following the order listed in Table 2.

once you copy a script in. put cursor on some of the text in the script and then

click on the "lighting bolt" in th etool bar. the one "WITH-OUT" the "I".

that will run the while script ; and as a result install all the procedures listed ion that file.

Table PICES Stored Procedures

|  |  |
| --- | --- |
| **Source File** | **Stored Procedure Name** |
|  |  |
| Classes.sql | ClassesRetrieveAll |
|  | ClassesRetrieveByName |
|  | ImageClassInsert |
|  | ImageClassDelete |
|  | ImageClassUpdate |
|  | ImageClassMerge |
|  | |
| Cruises.sql | CruisesDelete |
|  | |
| Stations.sql | StationsInsert |
|  | StationsUpdate |
|  | StationsDelete |
|  | |
| Deployments.sql | DeploymentLoadList |
|  | DeploymentLoad |
|  | DeploymentInsert |
|  | DeploymentUpdate |
|  | DeploymentDelete |
|  | |
| SipperFiles.sql | SipperFilesUpdateScanLinesAllFiles |
|  | SipperFilesCalcStats |
|  | |
| Images.sql | ImagesInsert |
|  | ImagesUpdate |
|  | ImagesEraseSipperFile |
|  | ImagesBuildConditions |
|  | ImagesGetClassStatistics |
|  | ImagesQuerySelectFieldNames |
|  | ImagesLoadByImageFileName |
|  | ImagesQueryByImageGroup |
|  | ImagesQuery |
|  | ImagesQueryByGrouop |
|  | ImagesQueryForScanLineRange |
|  | ImagesImageFileNamesByScanLineRange |
|  | ImagesUpdatePredictions |
|  | ImagesUpdateValidatedClass |
|  | ImagesUpdateInstrumentDataFields |
|  | |
| ImageGroup.sql | ImageGroupLoadAll |
|  | ImageGroupLoad |
|  | |
| ImageGroup.sql | ImageGroupDelete |
|  | ImageGroupEntryInsert |
|  | ImageGroupEntryInsertList |
|  | ImageGroupInsert |
|  | |
| ImageGroupEntries.sql | ImageGroupEntryInsert |
|  | ImageGroupEntryInsertList |
|  | |
| ImagesFullSize.sql | ImagesFullSizeLoad |
|  | ImagesFullSizeSave |
|  | ImagesFullSizeFileNamesForSipperFile |
|  | |
| InstrumentData.sql | InstrumentDataInsert |
|  | InstrumentDataGetByScanLine |
|  | InstrumentDataDeleteOneSipperFile |
|  | InstrumentDataLoadOneSipperFile |
|  | |
| FeatureData.sql | FeatureDataInsert |
|  | FeatureDataLoadByImageFileName |
|  | FeatureDataGetOneSipperFile |
|  | FeatureDataGetOneSipperFileClassName |
|  | FeatureDataGetOneImageGroup |
|  | FeatureDataGetOneImageGroupClassName |
|  | FeatureDataUpdateInstrumentDataFields |
|  | FeatureDataUpdateInstrumentDataFieldsForSipperFile |
|  | FeatureDataGetScanLinesPerMeterProfile |

## Client Installation.

Before you install the client software you need to have already performed the Server installation. You start the client installation by deciding where you want the PICES home directory to be. This will be the root directory for all Programs and Data Files that will be needed to support a client installation of PICES. Typically this is “C:\Pices” but can be anywhere that is a valid directory path. From this point the installation consists of four simple steps;

1. Setup the environment variable PicesHomeDir to a directory path that you want.

2. Run Setup Program.

3. Copy Over Needed Files.

3. Edit Configuration files, such as MySQ.cfg configuration file.

Details for each of these follow of the net couple of pages (Sections 2.2.1 thru 2.2.4 ).

### Set Up PicesHomeDirectory Environment Variable.

Before we can do anything we must decide where the programs and data files that comprise a PICES client installation are to be located. Typically this has been “C:\Pices” but it can be any valid directory path that you decide. The following description is for a “Windows 7” installation but other widows platforms are similar.

1. (Figure 1) Right click on “Computer” and then select the “Properties” menu item. This will bring up the dialog bog shown in Figure 2.

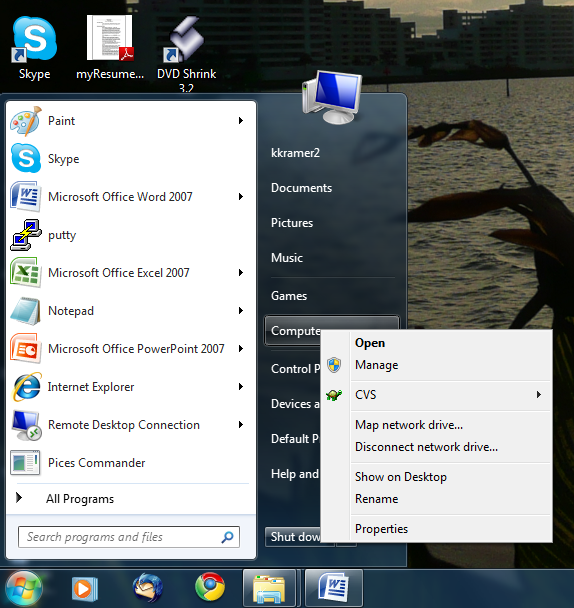


Figure Programs Menu. Right click on the "Computer" menu item to bring up menu.

1. (Figure 2) Click on the “Advanced System Settings” in the left pane. This will bring up the “Systems Property” dialog box shown in Figure 3.

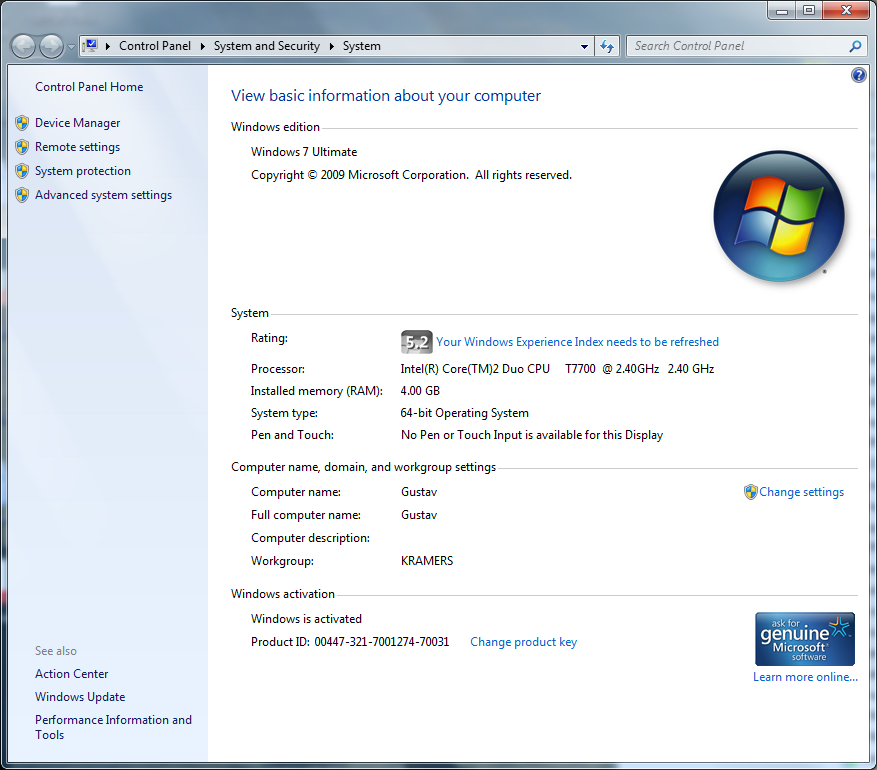


Figure Computer Properties; you will click on the “Advanced System Settings” button.

1. (Figure 3) Click on the “Environment Variables” button at the bottom of the dialog box. This will bring up the dialog bog shown in Figure 4.

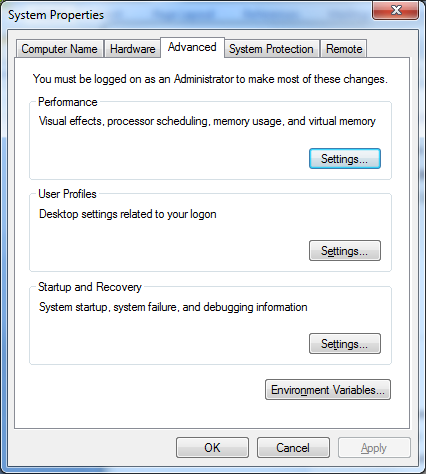


Figure Advanced System Settings.

1. (Figure 5) Under the “System Variables” section press the “New” button to add a new environment variable this will bring up the dialog box in Figure 5.

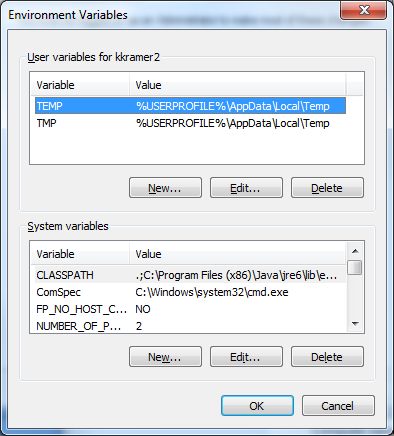


Figure Environment Variables. Under System Variables click on 'New'.

1. (Figure 5) We are now finally at the point where we can enter the environment variable. There are two fields in this dialog box. “Variable Name” and “Variable Value”. In the “Variable Name” field enter “PicesHomeDir”. In the “Variable Value” field enter the subdirectory where you want to install the PICES software. When you are done press the “OK” button. This complete the creation of the “PicesHomeDir” environment variable.

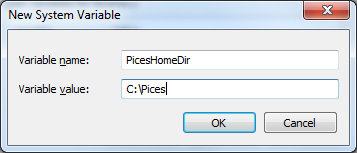


Figure 5 New System Variable. Enter data as above then press 'OK'.

### Run Setup Program.

There is a windows installation program that will install the PICES programs into the PicesHomeDir that you specified in the previous step (Figure 5). It will also install any windows components that are required for PICES such as “.net 3.5”.

1. Assuming you have the install CD in drive “D:”.
2. Make “d:\Pices\Setup” your current directory.
3. Run the SetUp program by typing “SetUp” followed by enter.

This will start the Client Setup program.

1. First dialog box will be “Welcome to the PicesSetup Setup Wizard” (Figure 6).

Press the “Next Button”.

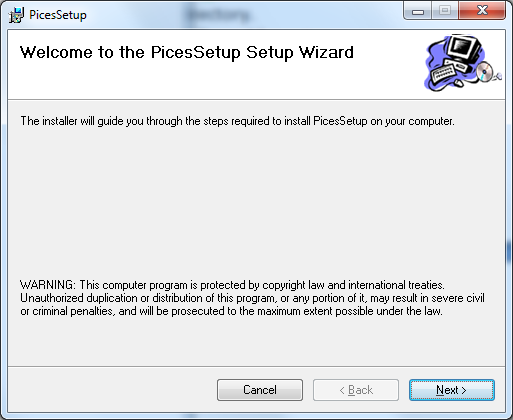


Figure 6 PICES Setup Program initial screen.

1. The “Select Installation Folder” dialog box (Figure 7) is where you specify where the PICES Software is to be installed. This needs to be the same directory where the Environment Variable “PicesHomeDir” was set to in step 2.2.1 on page 8. Set the “Folder” field to this directory name an click on the “Everyone” radio button at the bottom of the dialog box.

Press the “Next” button when you are done filling out the dialog box.

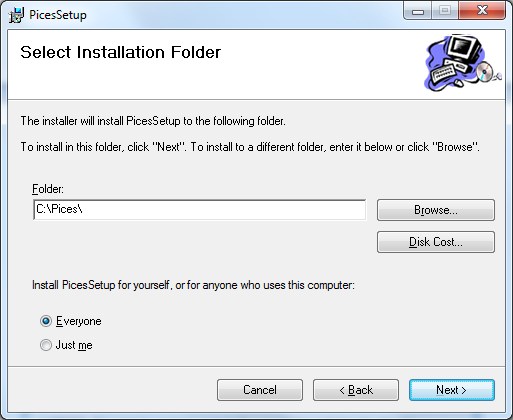


Figure 7 PICES Seup Select PICES Home Directory.

1. The “Confirm Installation” dialog box in Figure 8 gives you one last chance before installation starts. Press the “Next” Button to continue.

After you press the “Next” button installation will start; this process will take several minutes or longer depending on what additional widows components need to be installed.

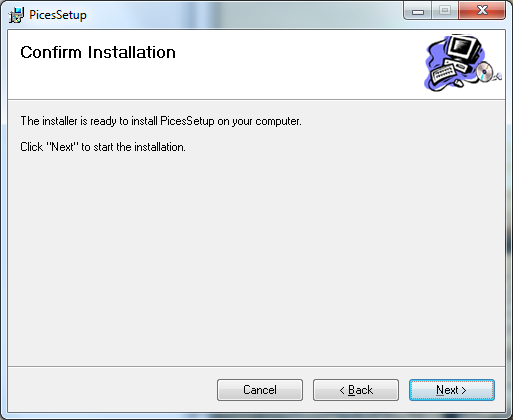


Figure PICES Setup Confirm Start.

1. After installation is completed you will be presented with one last dialog box “Installation Complete” as shown in Figure 9. Just press the “Close Button and the we are done with this part of the installation.

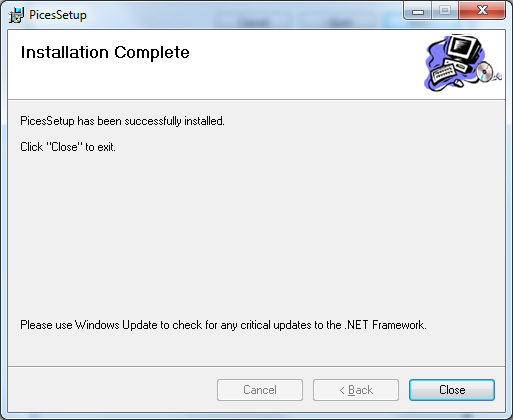


Figure PICES Setup Completed.

### Copy over Needed Files.

There are more files that will need to be manually copied over to the PicesHomeDir. Assuming that the install CD is in drive “C:” and that the PicesHomeDir is “C:\Pices” do the following steps. If they are different make adjustments as appropriate.

1. Open open a command window. (Get the DOS prompt). This can be done by typing “cmd” in the Search field of the start menu (see Figure 10). Then press enter. This will open up a DOS Command window (see Figure 11).

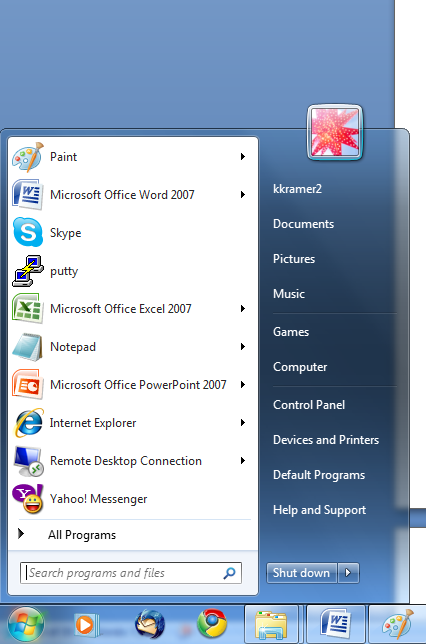


Figure Start Menu; Enter "cmd" in the 'Search program and fields' field.

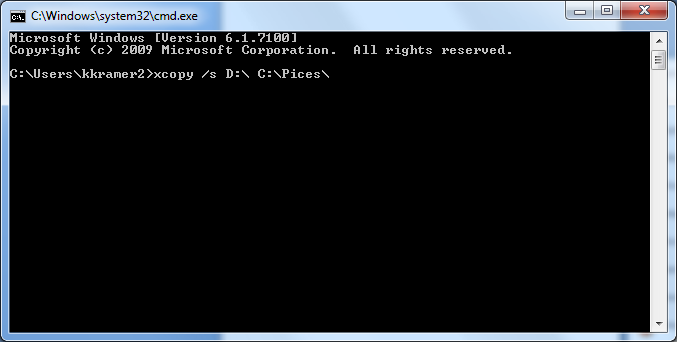


Figure 11 DOS Command window.

1. We are now going to copy over the files needs by PICES. Enter the line “xcopy /s d:\ c:\Pices” followed by the enter key. This will copy over all the data and configuration files needed by PICES

### Edit Configuration files, such as MySQ.cfg configuration file.

1. **MySql.cfg.**

You will need to know the name of several fields from the MySQL server installation. The technical support that installed the PICES database will be able to provide you with this information. The fields are:

|  |  |
| --- | --- |
| Description | Name that you want to give this particular PICES database installation. In general there is only need for one installation but it is possible to have more than one. |
| HostName | This will refer to the computer that is running the MySql software. This will typically be a IP/Address but can also be the name of the computer. |
| UserName | This is the MySQL User name created when installing the PICES database. It will need to have rights to all the tables and SQL Stored Procedures. |
| Password | Password assigned to ‘UserName’ |
| DataBaseName | The name of the Datbase assigned when created; typically will be “PICES” |

Figure 12 shows the contents of a typical MySql.cfg configuration file. There should already be one line that starts with the word “Server”. Each one of these lines is for a different PICES database. You will need to modify one of these lines to point to your PICES database server. Each field is separated by a ‘tab’ character. Each field has two parts, the name of the field and its value.

Assuming that your PICES dataset has the following values “HostName = 131.246.121.132”, “UserName = Kurt”, “Password = carrot”, and “DataBaseName = pices”. You will need to create a Server line as follows:

“Server Description:COT HostName: 131.246.121.132 UserName:Kurt PassWord:carrot DataBaseName:pices”

Remember that you need a ‘tab’ character between fields.



Figure MySql Configuration File.

Note the last line in “MySql.cfg” is “DefaultServer COT”. It specifies that the Server line with the description “COT” is the default database server to use. When a PICES application is started it will default to this database server. The user will be given a menu option to switch to a different server it they want.

1. **SipperDirectoryList.txt.**

This configuration file allows us to specify alternate locations for SIPPER files. By default PICES will assume that all SIPPER files can be found in the directory tree rooted at “${PicesHomeDir}\SipperFiles”. But given the size of the SIPPER files it is very likely that you will have them in more than one location. Figure 13 shows a typical SipperDirectoryList.cfg file. There are lines in this file; one for each subdirectory where SIPPER files can be found. When a PICES application such as PicesCommander starts up it will read this file for possible locations of SIPPER files and for each line it will recursively search the specified subdirectory tree for all SIPPER files. This will allow the application to build a directory that it can quickly search for known SIPPER files. Keep in mind that it is not necessary for the SIPPER files to be available; this is only if you want the ability to go back to the original SIPPER file when looking at a plankton image.

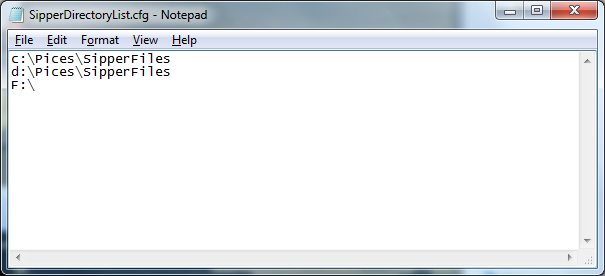


Figure SipperDirectoryList.cfg file; You can specify locations where SIPPER Files can be located.

If all your sipper files are going to be in the sub directory rooted at ${PicesHomeDir}\SipperFiles then there is not need to modify this file.

# Backing Up Database.

It is important to back up the Pices Database on a scheduled basis. This way if a problem occurs with the database that cannot be recovered from, such as a disk failure there is a means of recovering from a known point in time. Other times for backing up would be right after a significant amount of work has been done; ex: several thousand images have been manually validated; the extraction and classification of data from a cruise, etc. The MySQL database comes with utilities to facilitate a backup.

1) If you have not already done so install the MySQL Workbench application. Make sure it is at least version 5.22. You can find a download for the application at <http://www.mysql.com/>. The download is usually hidden at the bottom of the web page.

2) You will need enough free disk-space on an external drive to back up to. The amount of free space required will be dependent on the amount of data in the database. As a rough guide, if you have 20 million images you will need approximately 80 gigabyte of free space. 40 million images will require 160 gigabyte of space. These are just rough estimates based on previous experience.

3) Create a sub-directory on the external drive with the free space where you want to back up the PICES data to.

4) Start the MySQL Workbench application.

Figure 14 shows the screen that will first pop up. There are three different options to select from. We are interested in the “Server Administrator” functions that are circled in red below. In that section there should be a list of MySQL servers that can be administered. If the MySQL server that we are interested in backing uop does not appear you will need to create an entry for it at this time.

a. If an entry for your server does not exist press on the “New Server Instance” button.

1. Assuming you are running this program on the server computer select “locale computer”.

2. You will now be presented with Figure 15. The only field you will need to fill in is the default schema field. “Pices\_new”.

3. Click on the “next” button until you get to the dialog box shown in Figure 16. This is where you specify the type of MySQL Installation that you will be connecting to. In the case of USF Marine Science Pices Sipper-DB computer this would be “Windows (MySQL 5.1 x64 Installer Package)” You can now press the “next” button trough the next set of dialog boxes until you get to press the “Finish” button.

b. Select the server instance you want to back up.

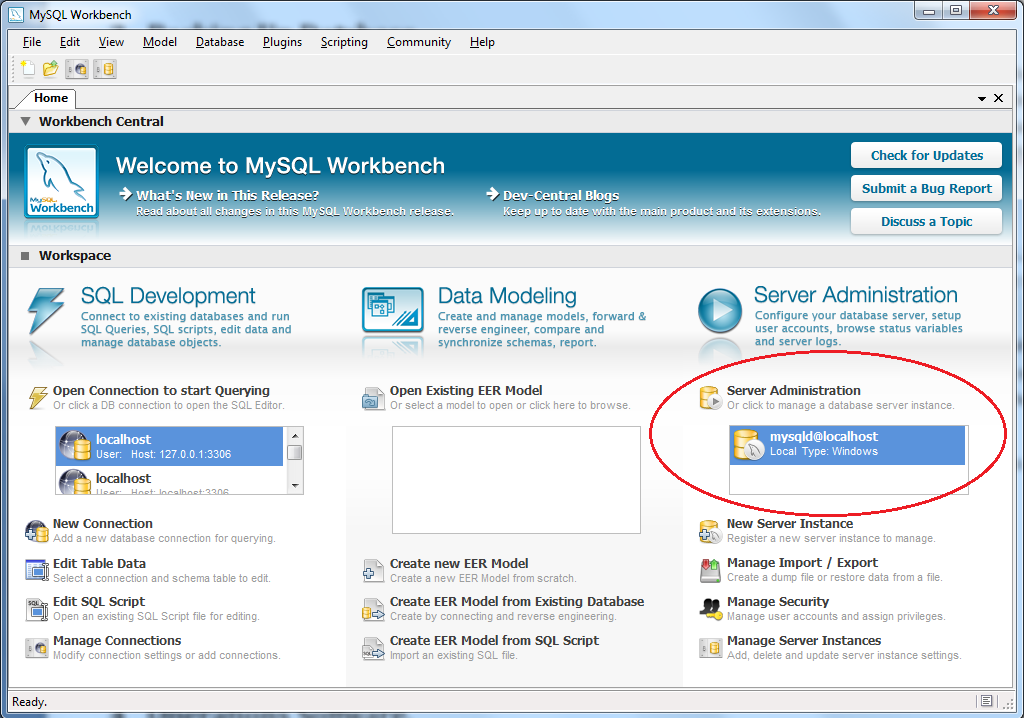


Figure 14 MySQL Workbench Main Screen.

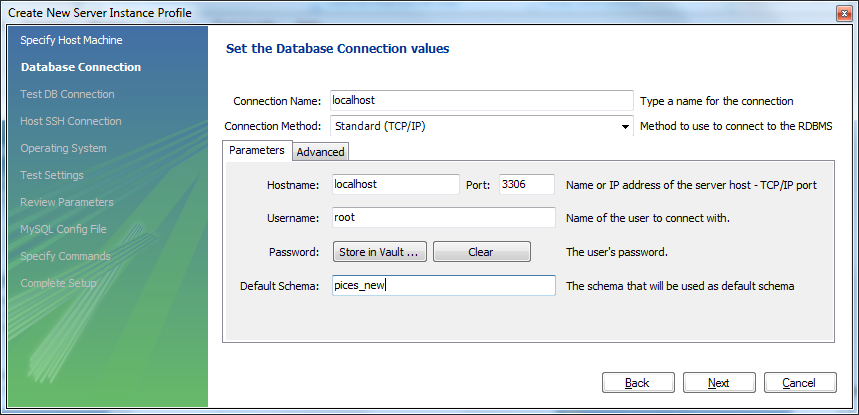


Figure Create a new server instance.

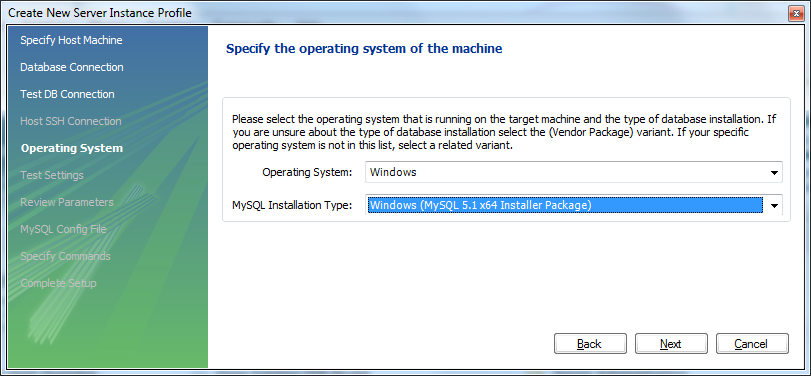


Figure MySQL Server Select Instalation Type.

5) You should now be looking at Figure 17.

a. At this time press the “DataDump” button that is circled in red below. This will bring up the fields that support back up of the databse.

b. Check the database that you want to back up. That is the item in the green box below assuming the name of the database is “pices\_new”. You will also need to click on the word “pices\_new” besides clicking on the check box.

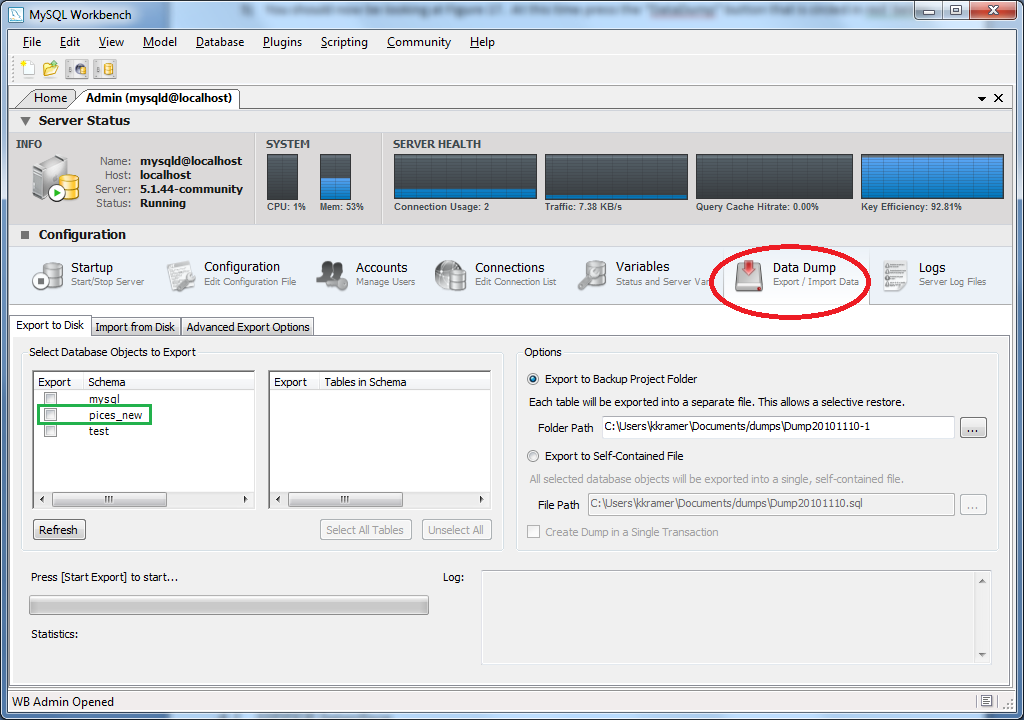


Figure 17 MySQL Workbench Administrator Window.

c. Figure 18 shows the screen just before you start backing up. You need to do two things.

1. Select the tables to back up. Do this by pressing the “Select the All tables” button in the red box in Figure 18.

2. Specify the directory to back-up to on the external drive by pressing the button in the green box in Figure 18.

3. Once you are confident that all is selected as you want press the button at the bottom of the dialog box called “Start Export”.

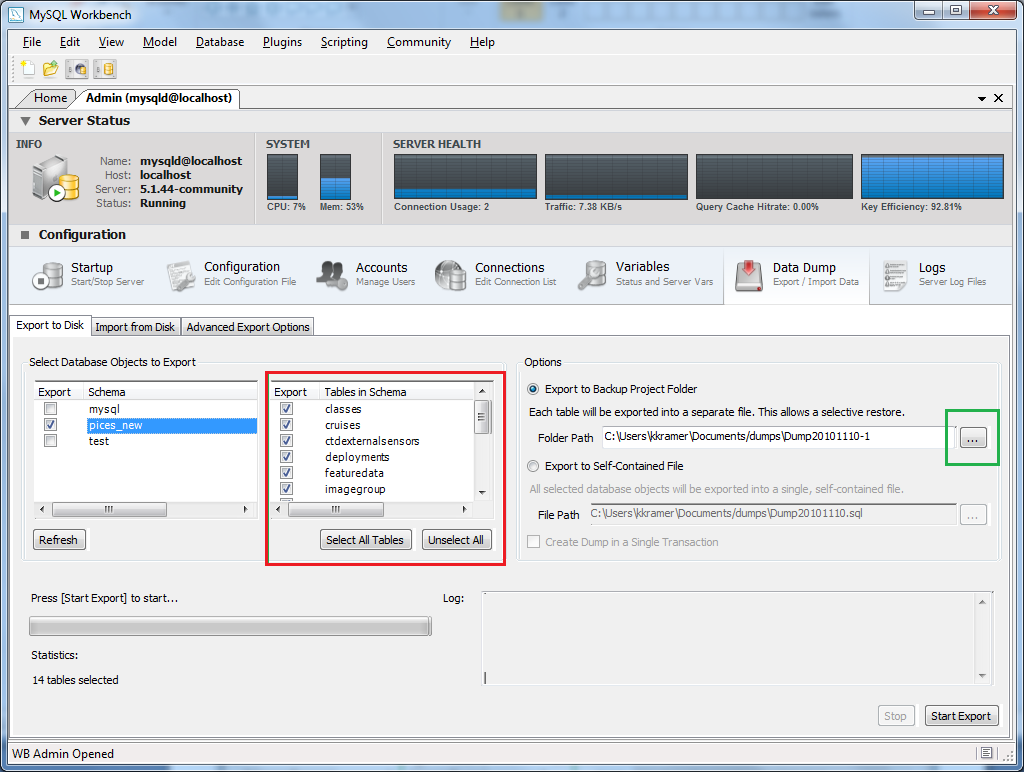


Figure MySQL Workbench Administrator Window; Database Selected; Ready to Start Backup.

5) There is no Dialog box that informs you that the Back Up is done. You will know that it is done because the “Start Export” button will show up again and be enabled.

a. Exit the MySQL Workbench application.

b. Disconnect external drive.

c. Ship external drive to offsite back-up location.

d. Offsite back-up location should immediately restore backup to back up server.

Back Up DONE.

# Restoring the Database

1st you need to restore the backup onto the laptop

[6:30:18 PM] Kurt Kramer: then you need to load the MySQL scripts

[6:35:26 PM] Kurt Kramer: 1) in WorkBench select Manage Import/Export

[6:35:38 PM] Kurt Kramer: 2) Select the database Server running on the Laptop

[6:35:46 PM] Kurt Kramer: 3) Select DataDump

[6:35:59 PM] Kurt Kramer: 4) Select the Tab "Import from Disk"

[6:36:42 PM] Kurt Kramer: 5) Seelect directory containing Backup of MySQL from other PC

[6:37:26 PM] Kurt Kramer: 6) Then I believe you press the Start Button

[6:37:44 PM] Kurt Kramer: after the restore is complete you will then need to reload all the MySQL scripts

[6:39:01 PM] Kurt Kramer: If you give me a couple of days I should be able to come up with a program that will install all the scripts for you and/or make sure they are up-to-date.

# Merging Pices Databases.

## Extracting from source database.

Run the following MySQL commands.

|  |
| --- |
| select \* into outfile "E:\\Dropbox\\Dropbox\\ToCotPices\\2015-06-23\\Classes.txt"  FIELDS TERMINATED BY '\t' OPTIONALLY ENCLOSED BY '"'  LINES TERMINATED BY '\n'  FROM Classes; |
| select i.SipperFileId,  i.ImageFileName,  i.ClassLogEntryId,  i.Class1Id,  i.Class2Id,  i.ClassValidatedId  from Images i where i.ClassValidatedId is not null  into outfile "E:\\Dropbox\\Dropbox\\ToCotPices\\2015-06-23\\ValidatedImages.txt"  FIELDS TERMINATED BY '\t' OPTIONALLY ENCLOSED BY '"'  LINES TERMINATED BY '\n'; |
| select \* into outfile "E:\\Dropbox\\Dropbox\\ToCotPices\\2015-06-23\\SipperFiles.txt"  FIELDS TERMINATED BY '\t' OPTIONALLY ENCLOSED BY '"'  LINES TERMINATED BY '\n'  FROM SipperFiles; |
| select \* into outfile "E:\\Dropbox\\Dropbox\\ToCotPices\\2015-06-23\\LogEntries.txt"  FIELDS TERMINATED BY '\t' OPTIONALLY ENCLOSED BY '"'  LINES TERMINATED BY '\n'  FROM LogEntries; |
| select \* into outfile "E:\\Dropbox\\Dropbox\\ToCotPices\\2015-06-23\\ImagesLogEntries.txt"  FIELDS TERMINATED BY '\t' OPTIONALLY ENCLOSED BY '"'  LINES TERMINATED BY '\n'  FROM ImagesLogEntries; |
| 1) “C:\Temp\PicesExportData\” make sure directory is empty from previous merge process.  2) Update stored procedure “ImagesExportClassificationData” to reflect the Last Date-Time that the procedure was used to merge data into COT - Pices  3) call ImagesExportClassificationData();  4) This will create export files in “C:\Temp\PicesExportData\” You will need to compress and move to appropriate place after. |
| select ValidatedClassName, count(ValidatedClassName) from ImagesLogEntries where DateTimeOccured > "2015-02-01 00:00:00" and (action = "V")  group by ValidatedClassName  limit 3000000; |
|  |
|  |
|  |
|  |
|  |
| Last Ran 2015-03-29 |

## Importing into destination database.

|  |
| --- |
| ImportClasses.py  a. Update Root Directory |
| ImportSipperFiles.py |
|  |
|  |
|  |
|  |

# Operations Software.

This is the software that allows the user to operate SIPPER in the field. There are two applications involved, “SIPPER Interface” and “SIPPER Disk Manager”.

* SIPPER Interface provides a GUI based interface for operating SIPPER. It provides functions for starting and stopping data recording, instrument data display, and real time display of SIPPER flight characteristics, such as Pitch and Roll. Other operations include data previewing, camera line display, and current Instrument Configuration Maintenance.
* SIPPER Disk Manager provides the tools for transferring SIPPER data off the SIPPER, initializing a SIPPER Drive, and formatting a SIPPER Drive.

## SIPPER Interface.

This is a GUI based application that is used when deploying SIPPER. It provides a friendly frontend to the terminal based interface that SIPPER provides. It provides the user with the means of specifying various configuration parameters, such as what device is plugged into which serial port, ex CTD is on serial port 2. It also provides graphical display of instrumentation data in both time and spatial (Depth Profile) domains.

## SIPPER Disk Manager.

This program gives the user the ability to manage the contents of a SIPPER disk drive. Since the SIPPER disk uses a proprietary format we need this program to allow us to view its contents, download the files onto a local computer, and reinitialize the drive for the next deployment.

# PICES Applications, Classification and Analysis Software.

The PICES application is a mixture of both GUI and command line based programs. The command line programs are designed such that they can run on Windows, Linux, and Solaris platforms while the GUI based applications are meant to run under the windows platform. There are some new tools that allow for the running of ‘.net’ applications under the Linux platform but this has never been tried with PICES software.

There is a general flow that you will follow while extracting and classifying data.

1. Deploy SIPPER
2. Download SIPER Files.
3. Extract and Classify Plankton images from SIPPER files.
4. View Plankton Images, judge results, and refine Training Library (Active Learning).
5. Re-Classify Plankton Images using refined Training Library from previous results.

## PICES Terminology.

**Class**: This term is used to designate a group of plankton images that share common attributes. You can define a group(Class) as coarse or fine as you wish. Examples of classes are “Chaetognath”, “Copepod”, and “Larvacean”. These are very coarse Class definitions. They can also be made to a finer “Copepod\_Calanoid”, “Copepod\_Macrosetella”, and “Copepod\_Oithona”. PICES supports a Class hierarchy such that you can specify a parent child relationship, and example Hierarchy being “Crustacean”, “Crustacean-Copepod”, “Crustacean-Copepod-Calanoid”. You can view this hierarch from PicesCommander by selecting the “Maintenance” and then “Plankton Class”, see Figure 24 on page 28.

**Training Library**: A training library is a collection of Plankton images from previous deployments of SIPPER arranged in a subdirectory tree. There will be one subdirectory for each class of plankton in this library. It is typically located off the PICES Home Directory (“${PicesHomeDir}\Pices\TrainingLibraries”). And example Training Library would be “etp\_2008” that came with the install CD. It contains 38 Classes arranged in 38 subdirectories.

**Training Model**: This often gets confused with “Training Library” it is similar but not the same. For a given Training Library there can be many Training Models. A Training Model specifies configuration parameters for the classifier, which classes to include in the classifier, level of granularity of classes (A few base classes or many refined classes), and other parameters. Referring to training library “etp\_2008” which came on the Install CD there are three included Training Models also included on the Install CD, “etp\_2008.cfg”, “etp\_2008\_Tuned.cfg”, and “etp\_2008\_Binary-Tuned.cfg”. Each one of those models uses different classifier parameters but the same Training Library to train from. For each Training Model there is a Configuration file. These Configuration files are located in “${PicesHomeDir}\dataFiles\TrainingModels\” directory.

**Bias Matrix**: Results from a 10 file cross validation of a Training Model. It is used by the Classification process (see 6.3.4on page 32) to adjust the finale results of a classification process to reflect the BIAS of the classifier.

**Classifier:** A Classifier predicts what class an unknown image belongs to. It is built using parameters from a specified from Training Model using labeled examples from the Training Models related Training Library to learn from. Once a Classifier is built(Trained) it can be used anytime in the future to make predictions. If you change the Training Models parameters or modify the related Training Library you will be to rebuild the classifier for the changes to take place. When a classifier makes a prediction it returns the Class that it believes is most likely correct.

**Validated Class**: Class assigned by the user(Expert) to a specific plankton image. In PicesCommander the user will have the ability to validate the class of any plankton image displayed.

**Break Tie**: This is the difference in probability between the two most likely classes that the classifier assigns to an individual plankton image. A “Break Tie” value of 0.5% indicates that the classifier finds very little difference between the two most likely classes.

**Active Learning**: This is a concept of reducing the number of images that the user needs to manually classify to achieve a desired level of classification accuracy. In PICES this is implemented by sorting classified images by “Break Tie”, low to high. The user would then manually classify the images that appear at the top of the list. Since these are the images that the classifier is having a hard time to distinguish they are the images that will most like have an impact on the decision boundary between classes.

**Image Groups**: PICES allows you to group images together by group. The most common use of this is when we harvest images randomly (section 6.3.6.2 page 33). It can also be used to group images that were imported from a sub-directory structure (section 6.3.6.3, page 34 ). PICES will allow you to View, Export, Classify, and Extract Feature Data by Image-Group.

## SIPPER File Viewer

**Sipper File Viewer**; Figure 19 shows the Sipper File Viewer application. This application provides the user the ability to display SIPPER files as they were recorded by the instrument. A typical SIPER file will be an image 10 cm’s wide by kilometers long with 10,000+ discrete plankton particles. The red box in Figure 19 is highlighting one specific plankton particle.

Figure 20 shows the selected image blown up in a separate window (Image Editor). In this window the classification results of two different classifiers are shown indicating that the particle is a “Chaetognath”.

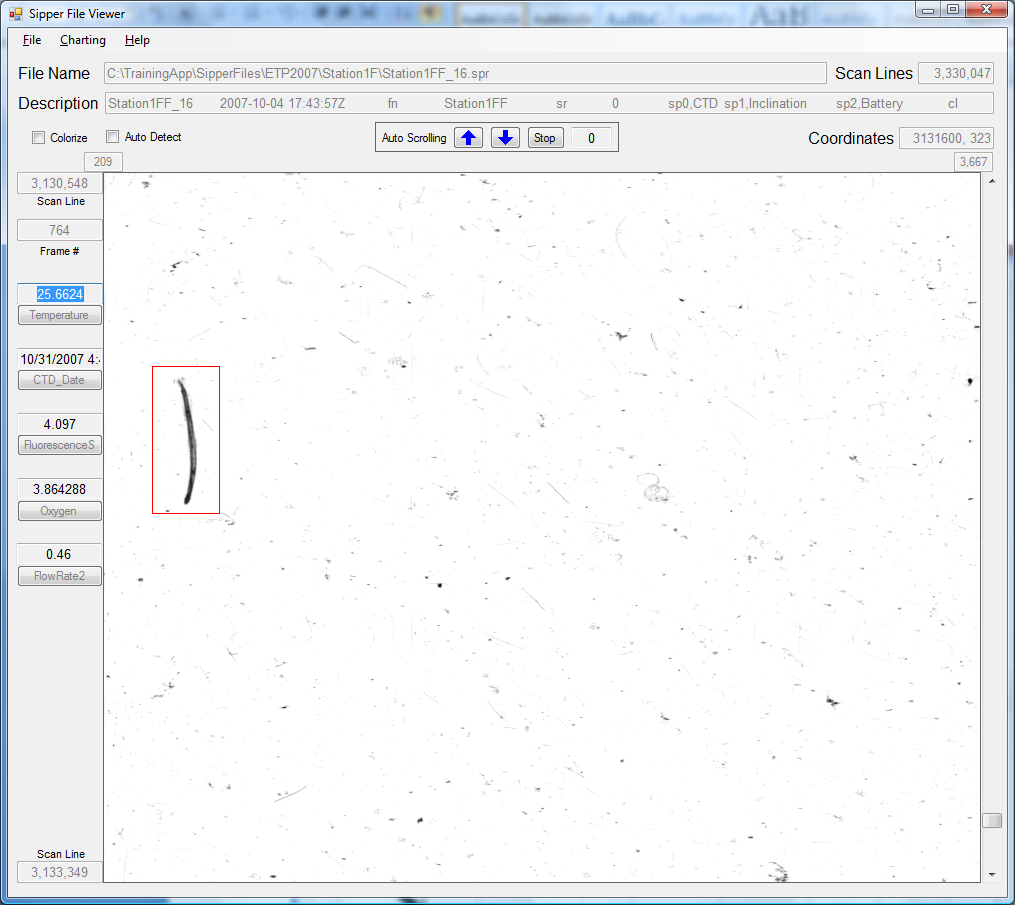


Figure Sipper File Viewer; a tool that allows user to view raw image data.

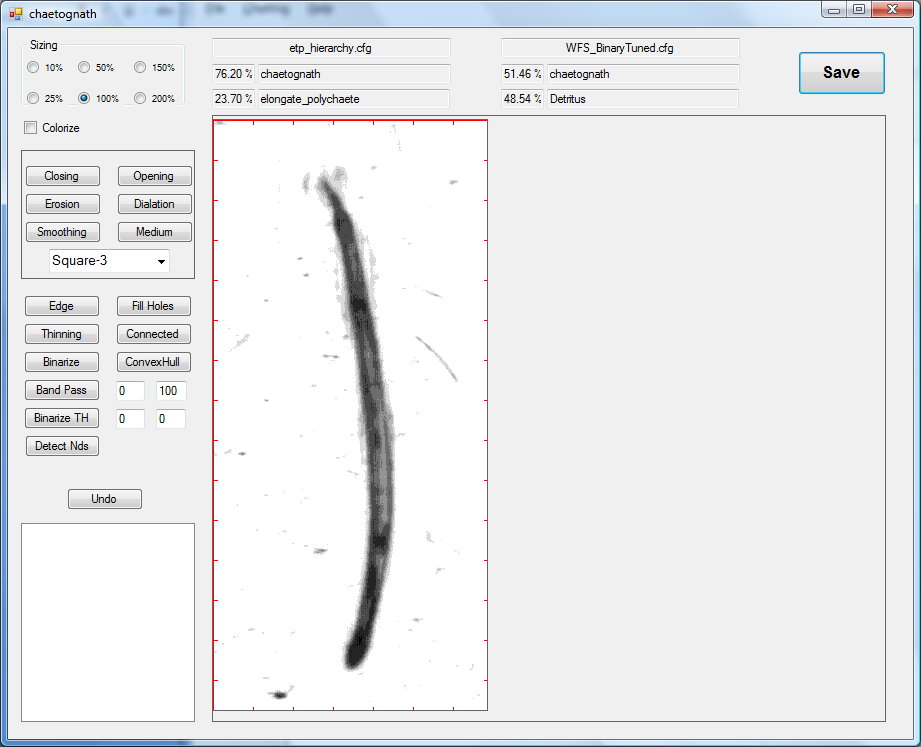


Figure Image Editor; Gives the user the ability to experiment with different image operations.

## PICES Commander.

### Introduction.

PicesCommander is meant to be both front end application to the PICES database and a means of performing tasks such Classification, Class Maintenance, Training Model Maintenance, and other items. Functions are provided in this application that will allow the user to create training libraries, classify images for a given, Deployment, Station, or even Cruise all at once producing classification reports with related statistics.

Figure 21 is a screen shot of the PicesCommander application showing images from the 2008 Panama Cruse, Station “1” Deployment “C”. The left panel provides statistics by class or depth, user selectable, which the user can then select to view from. The right panel provides a thumbnail view of all the images resulting from the current query. The user can choose to select a specific image from this thumbnail display (see Figure 23). There are many other functions that can be performed from this application that will be discussed below.

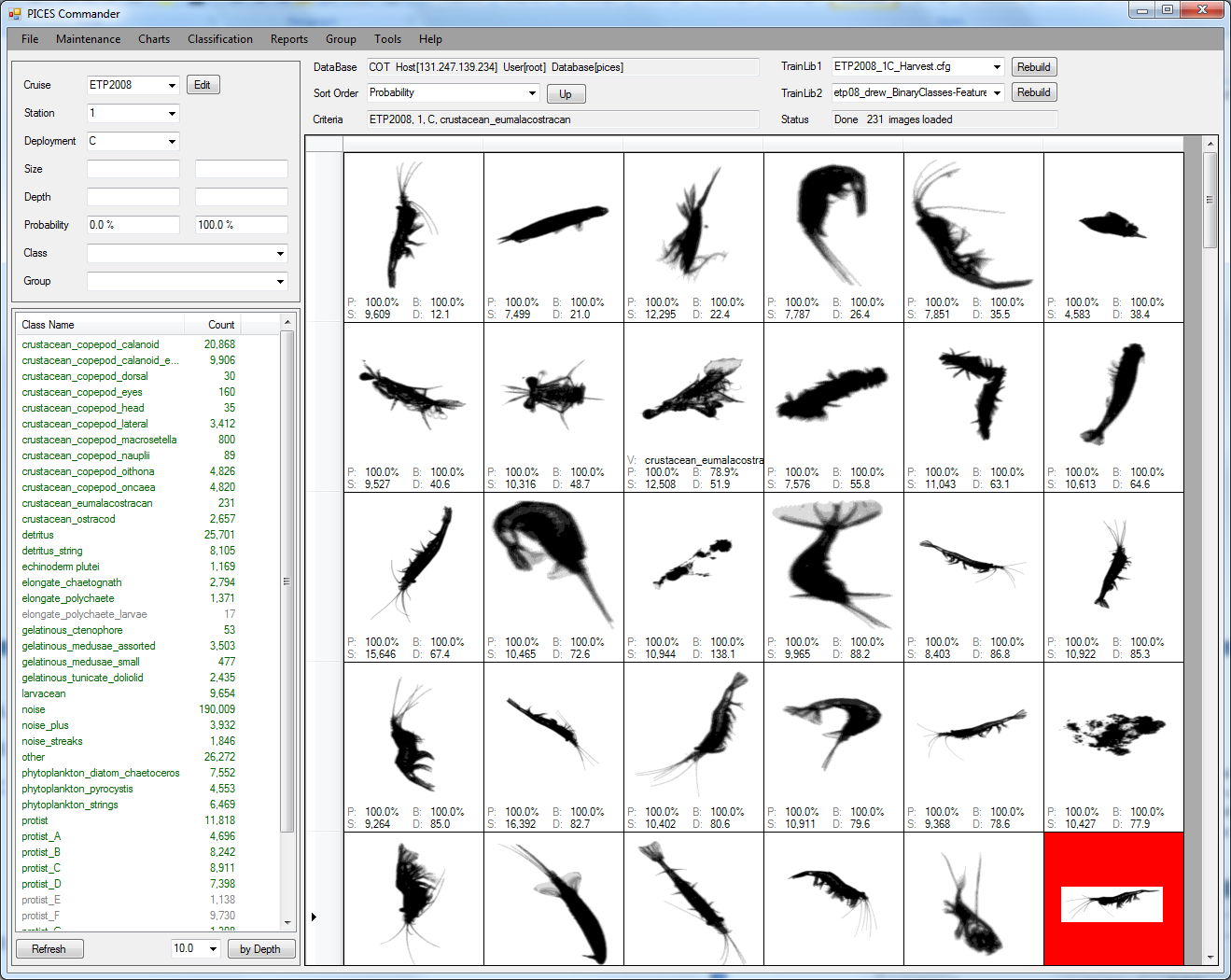
****

Figure PICES Commander Displaying shrimp from etp2008, station 1 Deployment C.

**Selection Criteria:** The upper left panel is where selection criteria are entered. This is where you specify what images you are interested in browsing. In Figure 21 images the “ETP2008” cruise, station “1”, and deployment “C” are selected. Other criteria that could have been selected are Size Range, Depth Range, Probability Range, and/or Image Group. Once you have entered your criteria you click on the “Refresh” button at the bottom left. When this button is pressed the program will start accumulating counts by Class or Depth and display them in the left panel. These counts will be continuously updated until all SIPPER files are read. By clicking on one of these entries the program will begin to load the related thumbnail images for all the Plankton images that meet the selected criteria for the Class or Depth selected.

**Thumbnail Display**: The main central panel displays Thumbnail images of the currently selected criteria. There is a field called “Criteria” just above the Thumbnail which display indicates the criteria that is currently selected. The thumbnail images can be sorted in one of several possible orders, Break Tie, Spatial, Class, Depth, Probability, and Size.

**Thumbnail menu Options**: By using the right mouse on any one of the thumbnail images a menu offering several options is presented. Figure 22 shows this menu being displayed for a selected image. You can choose to add this image to the currently selected Training Library, Validate its class by either selecting “Select Class” or one of the class names below it. Other options include copying either the Thumbnail or the full size image to the clip board to be pasted into another application such as word. This would be a good place and time to improve the current training library. This is accomplished by first validating the class and then adding it to the Training Library.

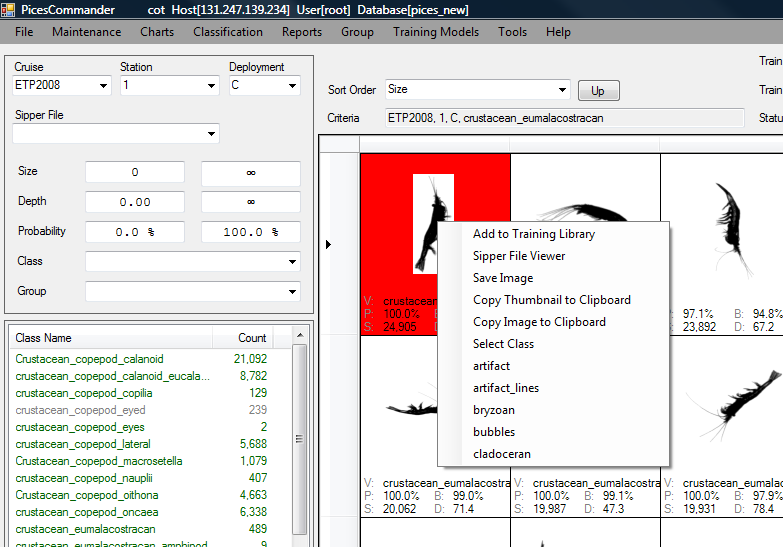


Figure PicesCommander Thumbnail Viewer, context menu.

**Viewing Image**: By double clicking on a Thumbnail image you can view the full size version of the image as in Figure 23. In this image you can see a full size version of the image along with classification stats from the two currently selected Training Models, instrumentation data is also presented. The instrument fields displayed can be selected by the user by double clicking of the instrument name label. The instrumentation data reflects the current instrument readings as of the time the plankton particle was imaged.

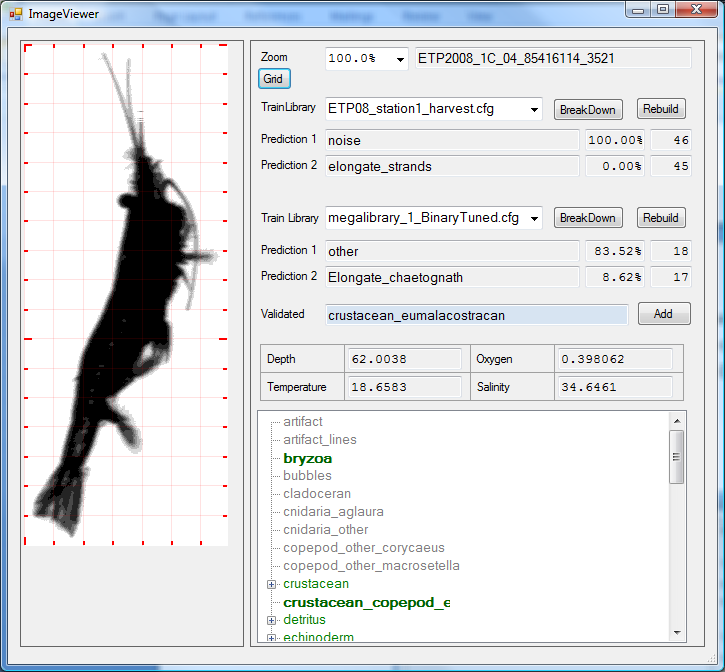
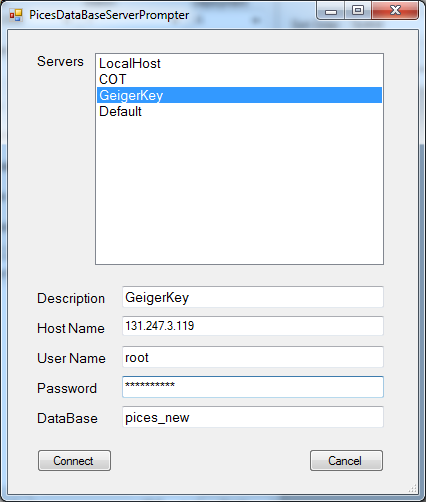


Figure Image Viewer, presents a full size view of image with classification statistics and instrumentation data.

### File Functions.

#### Connect to Database.

Use this menu option to select a different database server to connect to.



#### Save Images to Disk.

Using the current parameters in the selection criteria panel will save the appropriate images to a specified directory location. The images will be saved using one of two user selectable directory structures; by class or by depth. If by class images will be saved in the sub-directory with the same name as the images predicted class. If by depth will bin the images by depth using a user specified increment and save the images in the subdiretry that represents its depth.

#### Save Feature Data to Disk.

### Maintenance Functions.

There are several auxiliary tables that can be maintained from PicesCommander such as Plankton Classes, Cruises, Statons, Deployments, and SIPPER Files. These functions are reached from the maintenance menu option.

#### Plankton Classes.

Figure 24 shows the Class Maintenance function. From this function you can Create new classes, rename existing classes, delete existing classes, merge two existing classes into one class, and establish the Parent Child relationship. The left panel display all the current known classes reflecting the current class hierarchy.

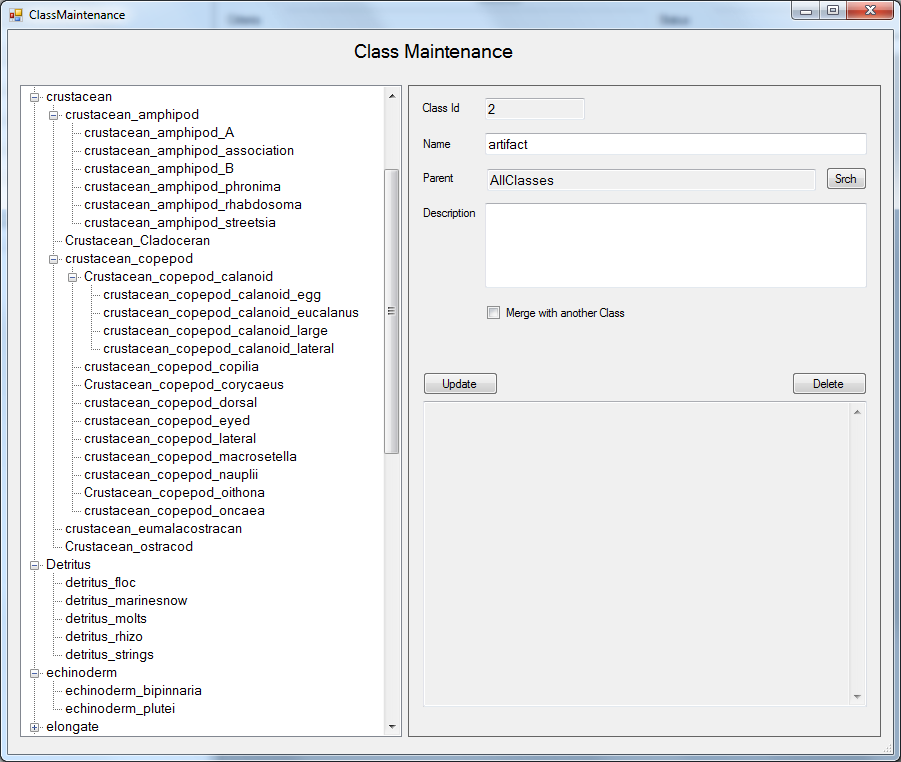


Figure Class Maintenance Function; Panel on right shows existing class Hierarch.

1. **Add a new Class**. In the left panel select the class that is to be the parent of the new class, if the new class has no parent then select the very root class called “AllClasses”. Press the right mouse button and a menu will pop up. Select the “Insert a new Class” option. This will cause the right panel to be blanked out with the Parent field filled in already with the selected parent. You just need to fill in the name of the new class and then press the Update button.
2. **Delete a Class**. In the left panel select the class you would like to delete, then press the “Delete” button in the right panel. This process can take a while depending on the number of images in the database. The application is going to scan all the images; any image that is assigned to the class being deleted will now be assigned to the deleted Class parent.
3. **Merge Classes**. Select one of the two classes that you want to merge. Then click on the “Merge with another Class” check box. This will cause a new field to appear, “Merge To”, that will allow you to select the other class that you want to merge with. When you press the “Update” button the “Merge To” class will be deleted, and all images that are assigned to the “Merge To” class will now be assigned to the class that you had originally selected. This process will take a while depending on the number of images on the database.

#### Cruise/Station/Deployment Maintenance.

Figure 25 shows the Cruise/Station/Deployment hierarchy maintenance function. From this dialog you can maintain the Cruise, Station, Deployment, and SIpperFiles database tables. This dialog box reflects the hierarchy or all the entries. For example in Figure 25 the highlighted line is of the first SIPPER file of deployment “B” done on 2009-dec-15th at station “1” on the Dec 2007 panama cruise. By pressing the right mouse button on this entry we bring up the dialog box shown in Figure 29. From this dialog box you can maintain the particulars of the specified SIPPER file.

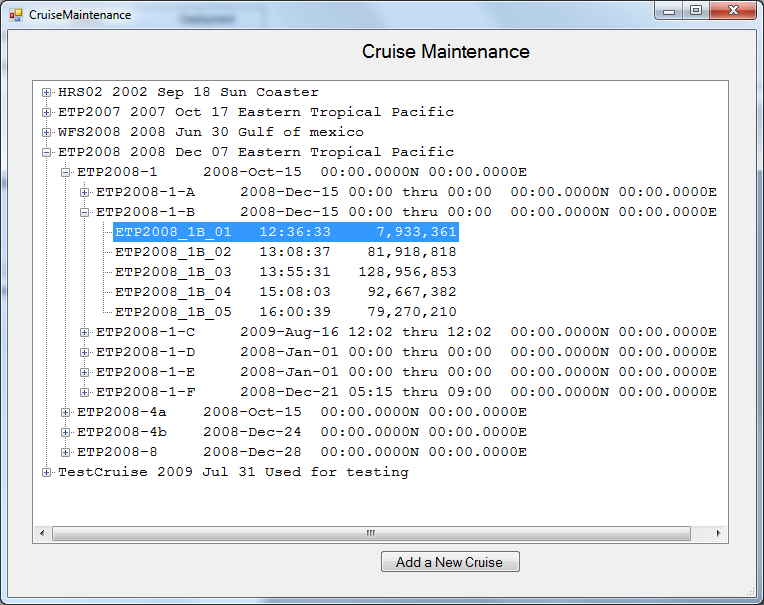


Figure 25 Cruise/Station/Deployment Maintenance Hierarchy.

**Cruise Maintenance:** This is where you maintain information about the Research Cruise and add entries for each Statoin that SIPPER will be deployed at.

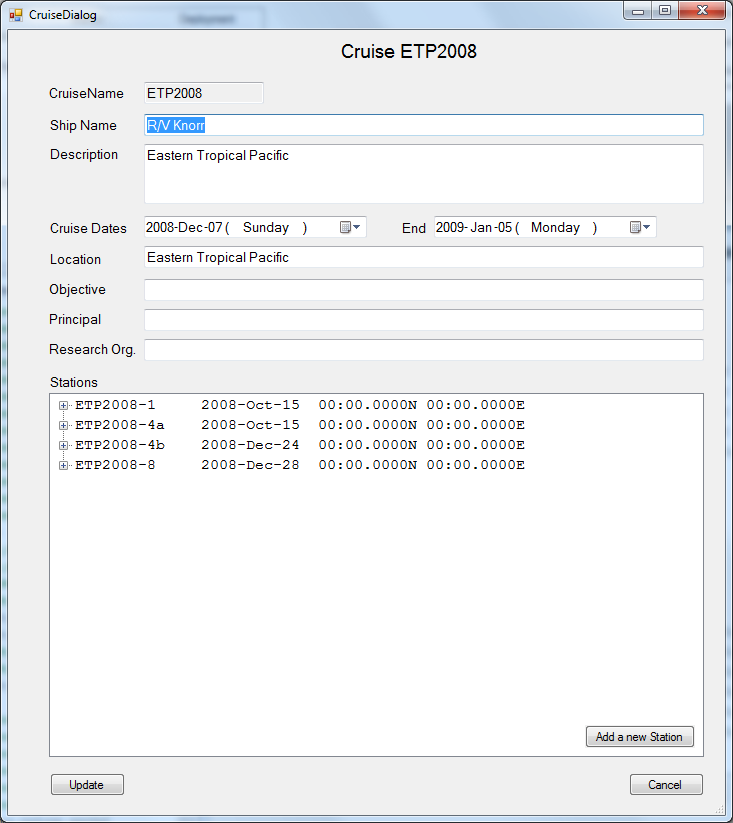


Figure 26 Cruise Maintenance.

**Station Maintenance**:

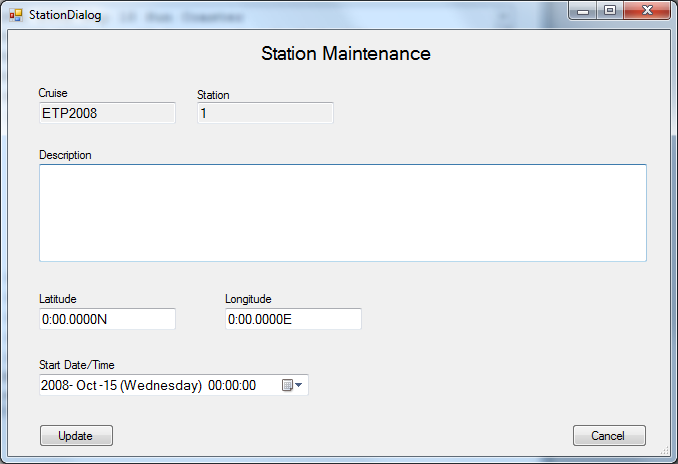


Figure Station Maintenance

**Deployment Maintenance**:

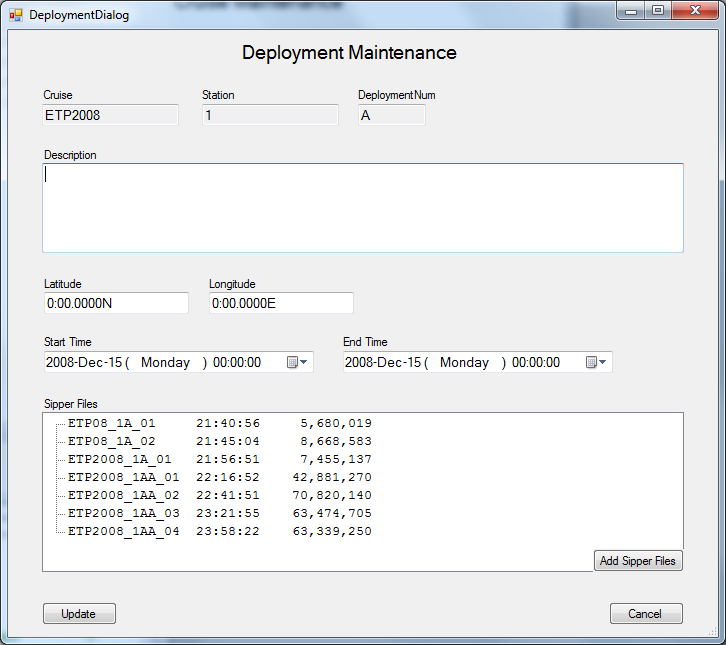


Figure 28 Deployment Maintenance.

**SIPPER File Parameter Maintenance**: Figure 29; this is where you can make modifications to SIPPER File parameters. These parameters specify items such as camera speed (scan lines per second), what instruments are attached to which serial port. What external instruments does the CTD device have attached? If these parameters are not correct the external instrument data will not be processed correctly.

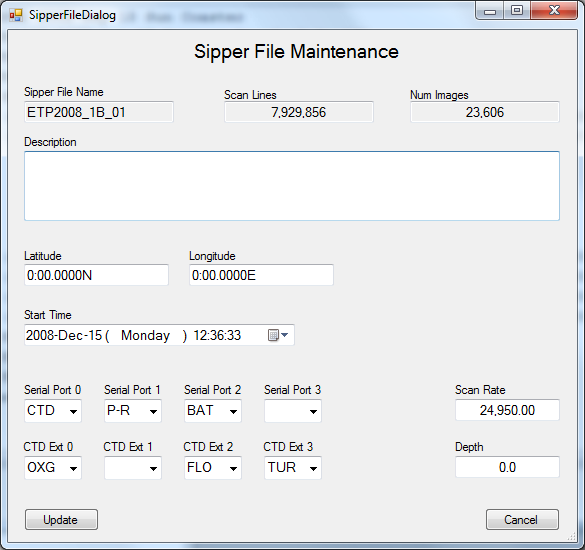


Figure 29 SIPPER File Parameter Maintenance

### Image Classification.

This function can be selected from the Classification Menu. This application will use the current selection criteria from PicesCommander to select images for classification. You will have the option to reclassify images or just perform a count based on current classification. You can classify at different levels of Hierarchy, Cruise, Station, Deployment, or SIPPER File. A classification report will be generated in the “C:\Pices\ClassificationReports” directory. This report will be a tab delimited text file that you can then load into Excel. The report will include result by class, by depth, and by probability. If a previous Bias Matrix existed for the specified classifier a set of results that reflect the known bias of the classifier will also be included.

BiasMatrix – This is a data file that can be created for a given Training Model. This is done by running a CrossValidation against the TrainingModel. This can be done from PicesCommander by selecting the “Ten Fold Cross Validation” submenu off the “Training Models” menu item.

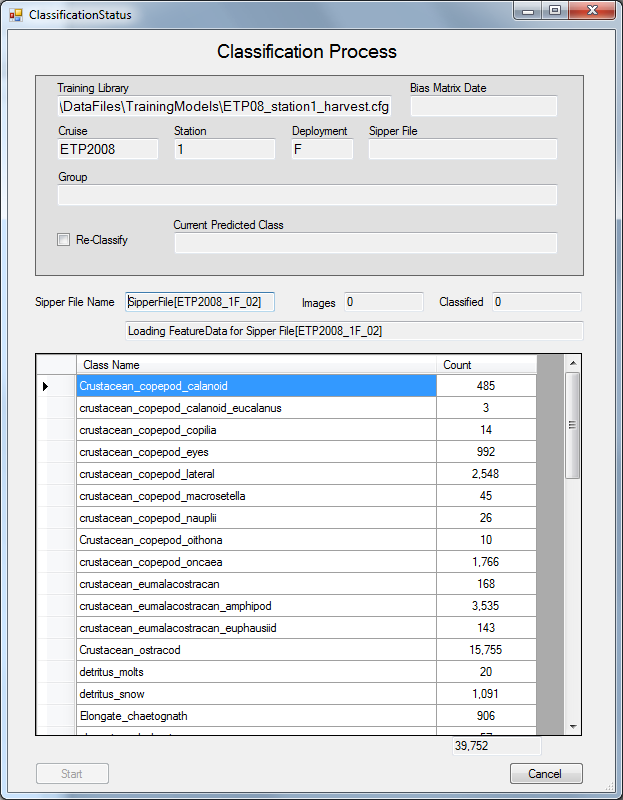


Figure Classification Process.

Two reports will be created:

|  |  |
| --- | --- |
| **Report** | **Depth Report** |
| ClassificationReport\_Cruise-SD01\_Station-02daysmall\_20130410-210020.txt | ClassificationReport\_Cruise-SD01\_Station-02daysmall\_20130410-210020\_Depth.txt |
|  |  |
| Size Distribution |  |
| Depth Abundance Distribution by Class 1 Meter | Depth Abundance Distribution by Class 1 Meter |
| Depth Density Distribution by Class 1 Meter | Depth Density Distribution by Class 1 Meter |
| Depth Abundance Distribution by Class 10 Meters | Depth Abundance Distribution by Class 10 Meters |
| Depth Density Distribution by Class 10 Meters | Depth Density Distribution by Class 10 Meters |
| Abundance Adjusted Results |  |
| Bias Adjusted Results |  |

### Create a Training Model.

In PicesCommander you can create a default Training Model from an existing Subdirectory. This is done by first preparing a subdirectory with training images arranged by Class. Each class will need to be in its own subdirectory. This function will then create a Training Model using default settings that includes all classes in the specified subdirectory.

### Abundance Adjustement.

1) It will only work for "Dual " classifiers.

In the [Global] section of the configuration file you need to define the “Other Class”. This is done by adding the line “OtherClass=<Name of Class>”; see the example of the Global section below. The Other class cannot be defined as any of the following “[TRAINING\_CLASS]” sections; but a subdirectory with the Other Class name must exist under the directory specified by the “ROOT\_DIR” in the “[Global]” section.

|  |
| --- |
| [Global]  MODELING\_METHOD = Dual  ROOT\_DIR = ${PicesHomeDir}\TrainingLibraries\GulfOilBroad2  Parameters= -Classifier1 GulfOilBroad2\_UsfCasCor.cfg -Classifier2 GulfOilBroad2\_MFS.cfg  Images\_Per\_Class=1000  Features\_Included=ALL  **OtherClass=Other**  [TRAINING\_CLASS] |

2) You need to run the "Abundance Adjustment Report" under the "Traninig Models" menu selection.

3) Then you can run the Classification report.

When computing statistics will weight each class equally; that is it will adjust for uneven class distribution.

When computing the Adjustment parameters the class distribution is normalized such that each class is treated equally.

### Group Assignments.

The PICES database allows you to maintain groups of images.

#### Group Maintenance.

Not implemented yet.

#### Random Harvesting.

This function creates a group of images by randomly selecting images from the PICES Database. It will use the current selected criteria in PicesCommander to select images from. The random selection will compensate for time at any given depth. The idea is that if a deployment spent a large amount of time at a given depth we do not want images from that depth to overwhelm the number of images at other depths. This was done so as to create Training Libraries that reflect the depth distribution of its various classes.

From my Dissertation:

The following description is from Kurt Kramer’s dissertation:

The examples for this dataset were randomly harvested from images collected at station 1 and then manually labeled by a marine biologist. The random harvesting of examples was done such that depth was weighted equally; that is, examples were grouped into 5 meter depth ranges and then examples were randomly selected from each depth range and weighted by the density of images at that depth. The idea is that the training library should reflect the underlying depth distribution of the different plankton classes. Since SIPPER was not deployed at various depths for equal amounts of time, the random sampling needed to be adjusted to reflect the density of images at given depth ranges. Figure 31 shows the algorithm used to randomly harvest.

Figure Random Harvesting Algorithm.

#### Import Group.

This function allows you to create a Group assignment from a subdirectory of images. You specify a subdirectory full of plankton images that are already in the Database. A new group will be created consisting of the images that are in this subdirectory.

#### Import Class Assignments.

The PICES database has a field called Validated Class in the Images table. This field indicates that a user/expert has specifically designates what class a given image belongs to. This function allows the user to assign the Validated Class to images in mass using a subdirectory full of images as input. The subdirectory name will be used to determine what class a specific image belongs to.

## Image Extraction and Classification

**Image Extraction and Classification**; A typical SIPPER deployment lasting several hours can generate a million or more images stored in SIPPER files as shown in Figure 19. Figure 32 shows the execution of the Image Extraction program. This program will process the SIPPER files; dicrete images are extracted, features calculated for each one; and classification of image is made. The results are stored in a database that can be queried later by the user through a application called PicesCommander (see Figure 21 on page 9).

There are two versions of this application that take the exact same options and produce the exact same results with th exception that one is Windows based and theother is command line based. Figure 32 is an example of the windows based version. It provides a visual dispaly of current run time stats. The specific avaialble options are listed below.

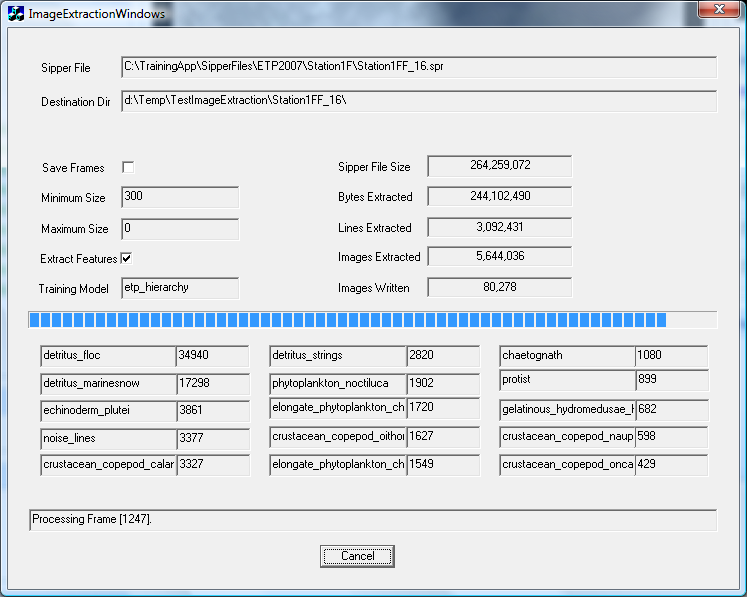


Figure 32 Image Extraction and Classification.

| Option | Argument | Description |
| --- | --- | --- |
| -s | sipper file name |  |
| -d | Destination Directory | Will builds a subdirectory below this directory. There will be one or more sub-directories for each class specified. |
| -r | Report File name | If left out will default to Sipper File name |
| -min | Num of Pixels | Minimum number of pixels a blob has to have to be extracted. Defaults to 250. |
| -max | Num of Pixels | Maximum number of pixels a blob can have to be extracted. |
| -morph | <o | c | d | e> | If specified will perform morphological operations on the raw data before extracting individual plankton images. Useful in very noisy environments. ‘o’ = Open, ‘c’ =close, ‘d’=dilation, ‘e’=erosion. You can specify any combination of operations. Ex: “C,C,O,O” will cause two opens followed by two closes. |
| -f |  | If specified, raw frames will be written to the destination directory. |
| -FramesOnly  or  -FO |  | If specified; only images frames will be written to destination directory; individual plankton images will not be extracted. |
| -sf | <1Bit | 3Bit | Sipper3> | Specify the SIPPER file format; if not specified will try to determine format automatically. |
| -c | Configuration File Name | The training model configuration file to use to build classifier from. If no directory path is included will look in default directory then in TrainingModels subdirectory under the PicesHomeDir. If this parameter is not specified then images will not be classified; in that case they will be labeled as Unknown. |
| -t | <Scan Line Num> | Allows you to start processing sipper file at a specific scan line. Useful when the start of the SIPPER file is full of noise which often happens at start of a deployment due to air bubbles in sampling tube. |
| -ipd | <Images Per Directory> | Maximum number of images that will be placed in a Sub Directory before a new one is created. Defaults to 2,000. |
| -NoDataBase  or  -NDB |  | If specified signifies that data is not to be written to a Database. If not specified the program will connect to the MySQL server specified tin the MySQL.cfg file. File is located in “$PicesHomeDir\Configurations” directory. |
| -A | Camera Number  <0 | 1> | SIPPER 1, had two cameras that were at right angles to each other. This parameter allows you to specify which you want to extract data for. This parameter serves no use for SIPER 2 or 3. |
| -Colorize |  | If specified the eight levels of gray scale will be assigned colors ratjer than a gray scale value. |
| -cpd | <num pixels> | Connected Pixel Distance. Distance in pixels that two pixels have to be within to be considered connected, default is 3. |
| -CountOnly  or  -CO |  | If specified individual plankton images will not be written to the destination directory. Generally used when you intent to store images in the Pices database. |
| -SaveAfter |  | If specified will same frame after pre-processing. Would also have to specify the “-f” option. |
| -Pre | <On | Off> | Specifies whether to preprocess from before extracting images. Defaults to “Yes”. |
| -X |  | If specified Feature Data will be extracted. This parameter must be specified if you want to classify images. |

## Grading Function (Command Line)

This is a command line application that will generate a confusion matrix that will indicate the accuracy of classification. If is meant to be ran after a classification is performed and the user has moved incorrectly classified images to a subdirectory that reflects their true class.

After a classification is performed; the classified images are placed into subdirectories where each subdirectory will represent a class. The subdirectory name will consist of the class name with a sequential number appended to the end (no more than 1000 images per subdirectory). A text file with feature data and classification data will be placed at the root directory of the classified images. The user can then move images from one directory to another to reflect there true class. What is important is that the name of the subdirectory they move it to indicate the class that it belongs to. Images can be moved by either dragging between directory via windows explorer or by the “Classify Border Images” function of the java gui app. All the images must be kept in the same subdirectory structure.

When the user is ready to grade the classification they can run the Grading application. This program will use the feature data file originally generated via the classification application as the source of the machine assigned class. It will then use the current location of each image to determine what the true class of the image is. The class will be determined by the name of the subdirectory the image is located in.

**GradeClassification**

-s <Source Directory>

[-r <Report File Name>]

[-H <HTML File Name>]

|  |  |  |
| --- | --- | --- |
| Option | Argument | Description |
| -s | <Directory name> | This is the directory that was the destination of the classification process. Under this directory the user would have moved images around to the subdirectories that reflect the plankton images true class. |
| -r | <Report file name > | Name of file to print report to. If not specifies will print to the command console. |
| -h | <HTML file name > | If you want a HTML compatible version of the report file |

The following is a conversation involving performing a grading of validated data using an existing training model. So you would say that we graded a Training Model against validated data.

[9:30:49 AM] Andrew Remsen: Do I need to run grading under the command line still?

[9:32:58 AM] Kurt Kramer: There is a GUI version;  but it only displays the results,  does not provide a text file.

[9:34:19 AM] Andrew Remsen: OK, but its not part of Commander. How would I use command line to grade a subset I created and validated? Do I to save to a directory first?

[9:35:37 AM] Kurt Kramer: You have a subset that you validatd in PicesCommander

[9:35:46 AM] Andrew Remsen: yes

[9:35:52 AM] Kurt Kramer: That will be the Test Data?

[9:36:00 AM] Kurt Kramer: what will teh traning data be ?

[9:36:20 AM] Andrew Remsen: ETP08\_BFS  
[9:36:22 AM] Kurt Kramer: you want to use one of the ok

[9:36:27 AM] Kurt Kramer: ok

[9:36:41 AM] Kurt Kramer: then there are two different commands you can  use

[9:37:01 AM] Kurt Kramer: Give me a min to get th command lines

[9:43:06 AM] Kurt Kramer: GradeClassification  -c ETP08\_BFS  -gt <Specify Directory where ground truth images are>  -R <Name of report file>

[9:43:46 AM] Andrew Remsen: OK. So I do need to create a local copy of all images in a dir?

[9:43:59 AM] Kurt Kramer: yes

[9:45:02 AM] Andrew Remsen: Is that export images?

[9:45:46 AM] Kurt Kramer: yes (Pices Commander, Tools Menu, Export Images)

[9:47:12 AM] Kurt Kramer: 1) In pices Commander Select the criteria that bring up the images that you want.

[9:47:27 AM] Kurt Kramer: clicking on the Validated radio button

[9:47:51 AM] Kurt Kramer: 2) Select "Export Images" from tools menu.

[9:48:18 AM] Kurt Kramer: 3) Select Subdirectpry where to put images

[9:48:31 AM] Kurt Kramer: 4) Select "Class" for directory structure

[9:48:46 AM] Kurt Kramer: click on start.

[9:48:54 AM] Andrew Remsen: thx

[9:49:45 AM] Kurt Kramer: The GradeClassification program will generate a plane text file and a html file.

[9:50:39 AM] Andrew Remsen: I am gonna compare against your grading. And see how proprtional representation in the subset grades out against a more equitable training library

[9:50:39 AM] Kurt Kramer: It will also ghenerate a confusion matrix by hierarchial level.

## Grading Function (GUI)

**Grading Function**; Figure 33, Figure 34, and Figure 35 show the Grading Function in action. This application measures the performance of a specified training library against either a Ground Truth or by running a 10 fold cross validation. Figure 33 is where both the Training Library and Ground Truth are specified.

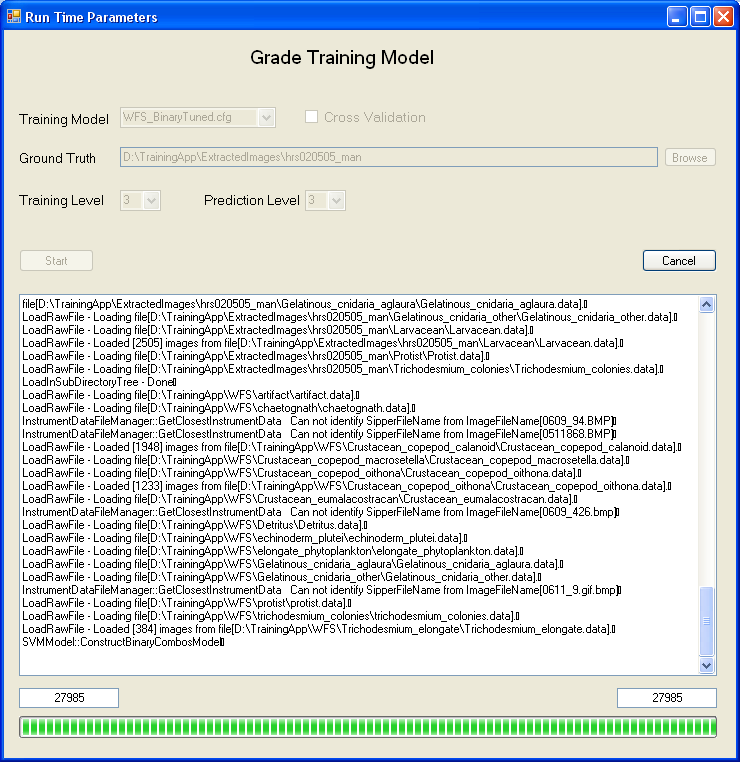


Figure Grading Function parameter selection.

Figure 34 shows the results of the grading. The display is in a cross validation format. The user has the ability to select any intersection and see what images had shown up there. Figure 35 shows the results of one such intersection. In this case three Trichodesmiums were classified as Chaetognaths. In this display a complete breakdown of all the binary classifiers involved in the prediction are include. This is done to provide a tool to help understand how certain classification errors come about and help adjust the Training Library in the future to not make this mistake.

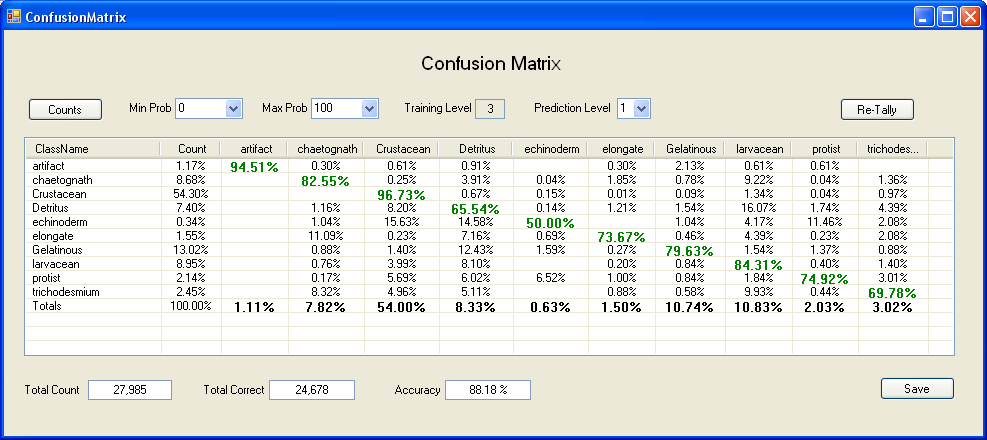


Figure Grading Function Confusion Matrix

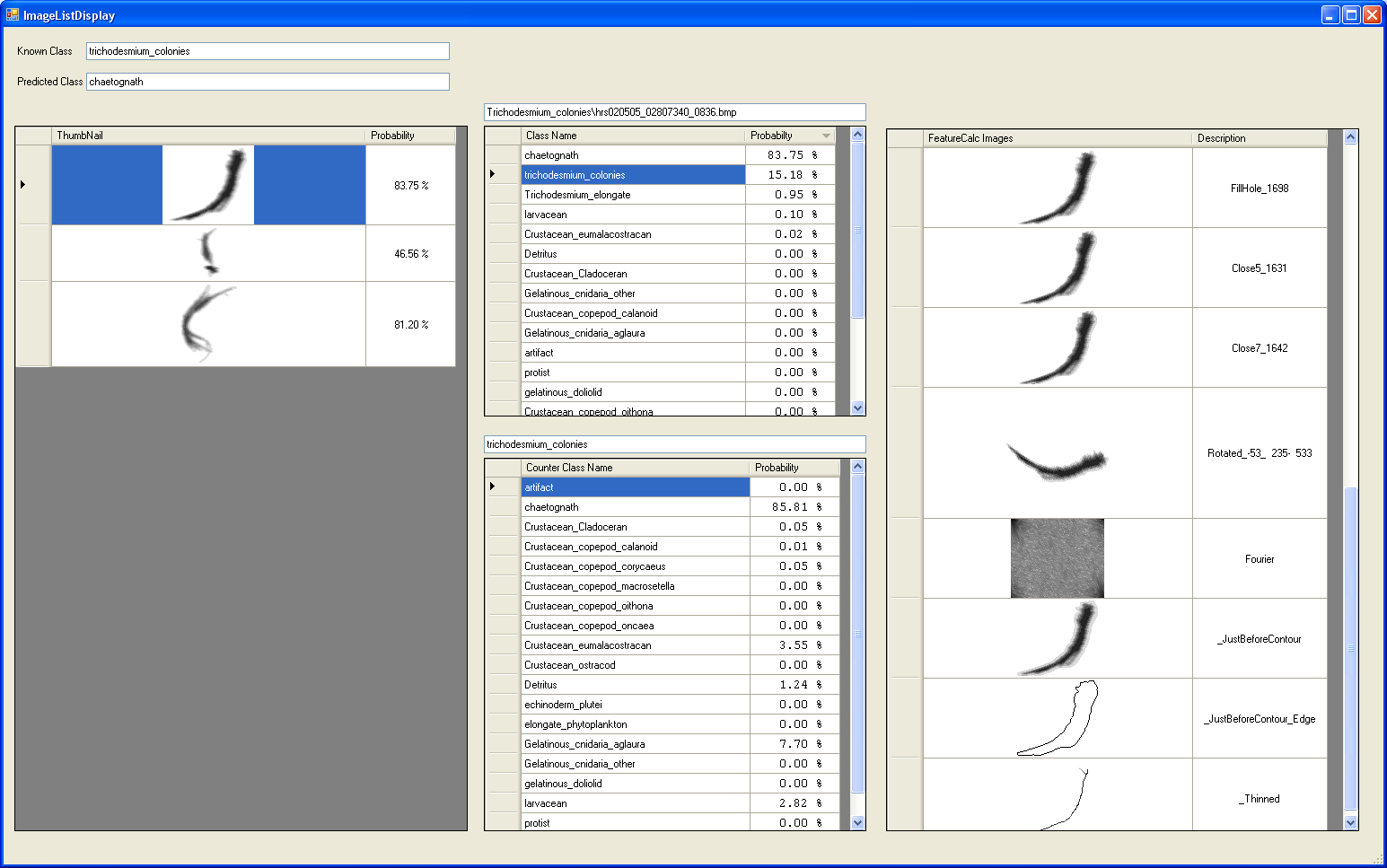


Figure Grading Function Displaying breakdown of classification of single images.

Definitions to help interpret results in report.

Total-Examples: Total number of predictions; same as “Total-Images”

Part-of-Classifier: True if: Known class is (Defined as Other Class) or (any class in the classifier starts with the same name).

Ex: If classifier contains “Crustacean\_Copepod” and prediction is “Crustacean” then would be considered part of classifier.

Total-Part-of-Classifier: Total number of examples that are “Part-of-Classifier”.

Noise-Class: Class considered noise if its highest level is one of the following: "UNKNOWN", "UNDEFINED", "NONPLANKTON", "NOISE"; ex: “Noise-Lines” would be considered a noise class.

Known-Class: The class that an example was ground trothed as.

Percent-Part-of-Classifier: Total number of predictions that are considered ‘Part-of-Classifier’ divided by ‘Total-num-of-Predictions’.

Total-Noise: Total number of examples where the “Known-Class” was considered “Part-of-Classifier” and “Noise-Class”.

Percent-Noise: Fraction of All-Examples that are of a “Known-Class” that is flagged as “Noise-Class”.

Total-Non-Noise: Total number of examples where the “Known-Class” was considered “Part-of-Classifier” and NOT “Class-Considered-Noise”.

Percent-Non-Noise: Fraction of All-Examples that are of a Known Class that is NOT flagged as “Noise-Class”.

***Note: “Total-Noise” + “Total-Non-Noise” = “Total-Part-of-Classifier”***

***Note: “Percent-Noise” + “Percent-Non-Noise” = “Percent-Part-of-Classifier”***

Total-Correct: The number of predictions that were correct; note that when utilizing Dual classifier there can be partial credit when only part of the predicted class equals part of the known class.

Accuracy-All: Total-Correct / Total Predicted.

Accuracy-Part-of-Classifier: Of the examples where known class was “Part-of-Classifier” the number that were predicted correctly / total number that were considered “Part-of-Classifier”.

Total-Non-Noise-Correct: Number examples that are NOT a “Noise-Class” and “Part-of-Classifier”

Total-Noise-Correct: Number examples that are “Noise-Class” and “Part-of-Classifier”

Accuracy-Non-Noise: “Total-Non-Noise-Correct” / “Total-Part-of-Classifier”

Accuracy-Noise: “Total-Noise-Correct” / “Total-Noise”

## Image Classification (Command Line)

This is a command line application that will classify images from a specified source directory using a specified training model and move them to a destination directory where they will be arranged by subdirectory according to their class. These images will typically be extracted from a SIPPER file. Arguments enclosed in square brackets [ ] are optional while angle brackets < > indicate that they are mandatory. A similar function can be performed in the PicesCommander program (Section 6.3, page 24)

**ImageClassificationApp.exe**

-c <Training Model>

-d <Destination Directory>

-s <Image Source Directory>

[-b <Border Images Directory Name>]

[-n <feature file name>]

[-r <report file name>]

|  |  |  |
| --- | --- | --- |
| Option | Argument | Description |
| -b | <Directory Name> | The directory in which all borderline images will be placed. A borderline image is an image that the classifier could not make a definitive decision on during classification. |
| -c | <Training Model File> | **REQUIRED** The training model configuration file. This file informs the program what classes to attempt identification for. |
| -d | <Destination Directory> | **REQUIRED** The directory to which the extracted images will be written. |
| -n | <feature file name> |  |
| -r | <report file name> | If not specified, then the program will default to SIPPER-File-Name, with '.txt' as the extension, and placed in the destination directory. |
| -s | <image source directory> | **REQUIRED** The base directory that contains the images to be classified. |

## Spatial Analysis

## Our Neighbors

A program that produces a report of the nearest neighbor distances between particles in a SIPPER file. It can be used for all Classes or for a specific class. The source of data is a directory that consists of classified images such as one produced by the classification application. Given a directory path the program will process all subdirectories below it. The name of each sub directory determines the class name of particles(Image Files).

**Ourneighbors**

[-BucketCount <Number of Buckets in Histogarm>

[-BucketSize <Size of each bucket>

[-FromPlankton <Specific Plankton to process>

[-Iterations <Number of random Iterations>

[-NNT <Any | Same>]

[-Report <Report File Name>]

[-Random]

[-Recalc]

-Src <Source Directory>

|  |  |  |
| --- | --- | --- |
| Option | Argument | Description |
| -BucketCount  or  -BC | <BucketCount> | Only used when ‘–Random’ specified. A integer greater than 0 that represents the number of buckets in the Histograms in the Random Report. If ‘BucketSize’ and ‘BucketCount’ are not provided then will default to 100. If just ‘BucketCount’ is left out then will be computed using the largets Nearest Neighbor distance divided by BucketSize. |
| -BucketSize  or  -BS | <Bucket Size> | Only used when ‘-Random’. A integer greater than 0. Specifies the size of each bucket in the Random histogram reports. If not specified then will be computed for each class: BucketSize = (Largest Nearest Neighbor distance divided by BucketCount) |
| -FromPlankton  or  -FP | <IPlankton Class> | Optional, if specified then the reports will be for on the the specified Plankton Class |
| -Iterations  or  -I | <Num of Iterations> | Only used when ‘-=  . If left out will default to 20. |
| -NearestNeighborType  or  -NNT | <Any | Same> | Defaults to Any  Any – Look for nearest particle regardless of class  Same – Look for Nearest particle of same class. |
| -Report  or  -R | <Report File Name> | Defaults to console. If ‘-Random’ specified then there will be additional report files, one for each class. The names of these files will consist of the specified Report File Name plus the class name. |
| -Random |  | Optional, if specified, Random reports will be generated for each class. |
| -Src | <Directory> | Mandatory, Specify the root directory where the classified images are. This would typically be the result of a Image Classification process. |

=======================================================================

OurNneighbors -Src “C:\TrainingApp\ClassifiedImages\hrs020304”

-Report Hrs020304\_Neighbors.txt

-Random

-BucketCount 200

-BucketSize 50

-Iterations 500

=======================================================================

OurNneighbors -Src “C:\TrainingApp\ClassifiedImages\hrs020304”

-Report Hrs020304\_Neighbors.txt

-Random -BC 200 -BS 50 -I 500

=======================================================================

## Delete Duplicate Images

There are instances where the same plankton image is in the training directories more than once; there are three sources for this situation. 1) Sometimes the same image is copied into two separate directories representing two different classes. 2) A bug in an earlier version of the SIPPER device would cause the contents of the last image buffer to be written twice to the end of a SIPPER file. 3) Naming conventions changed several times earlier in the project, so SIPPER files that were extracted before and after the naming convention change would generate identical plankton images with different names.

There are two types of duplicate situations to worry about. One is where the same image with two different names is placed in the same training directory. This causes that one particular sample to be weighted twice. The other problem is that the same plankton image was placed into two different training directories. This can cause very serious problems when both training directories representing two different classes is used in a model.

To deal with the issues listed above a function was developed that would detect duplicate images. There are two ways in which duplicates are detected. The first method is to look for images that have the same name. In this case the plankton image would have had to be in two different directories representing two different classes. The other detection method looks for plankton images that have identical calculated features. In this case the duplicates can be either in the same directory of two different directories.

When in the same directory it is reasonable to keep one of the images while discarding the other. The user has on two occasions confirmed that the image is of that class. When the plankton images are in two different directories representing two different classes then there is a serious issue to be dealt with. In this case the user has chosen at different times that the same image is of two different classes. In this case the safest action to take is to remove both images.

This function is used in three places, the training process when classifying and a command line program that will remove duplicates from a specified directory structure and place them into a temporary directory for user review.

**DeleteDuplicateImages**

[-RootDir <>]

[-ReportFile <report file name>]

|  |  |  |
| --- | --- | --- |
| Option | Argument | Description |
| -RootDir | <Directory Name> | This is the root directory of the sub directory structure that you wish to search for duplicates. If not specified then will default to the current directory. The program will recursively search through this directory and all its subdirectories. |
| -ReportFile | <Report File Name> | Name of file where a listing of all duplicate images located and where they were found will be printed. If not specified will default to the name of the RootDir parameter with an extension of “.txt”. |

When duplicate images are detected in the same directory but with different names the one that comes from an earlier part of the SIPPER file will be retained while the other will be removed. When the duplicates are in different directories then both will be removed.

When a duplicate image is removed it is actually moved to the directory “C:\Temp\DuplicateImages\” with the file name “<Name of image kept>-<Name of image removed>.bmp”.

## Cross Validation.

A application that will perform a K fold cross validation on a given set of feature data. Typically you would do this against a training model to test particular configurations [performance.

**CrossValidation**

-Config <Configuration File>

[-DF <Feature file>]

[-Folds <Number of folds>]

[-Format <Feature File Format>]

[-Hierarchy]

[-Log <Log File>]

[-MissClassed <Directory to copy Mis-Classed

images to>]

[-Normalize]

[-Randomize]

[-Recalc]

[-ReportNorm <report File Name>]

[-Save <File name to save to>]

[-TryEachFeature]

[-Validation <Validation data File>]

|  |  |  |
| --- | --- | --- |
| Option | Argument | Description |
| -Config  or  -C | <File Name> | Name of the Configuration file to use, including directory path where to locate. This file will define where data is to be taken from, parameters to be used by the classifier, features, etc. If the –DF option is also specified then the actual feature data will come from the data file father than the training libraries. |
| -DF  or  -DataFile | <Feature Data File> | You specify this option if you want to use feature data from the specified file rather than using the training libraries. You can use this in conjunction with the –Save option. In an earlier run you us the -Save option to save the Feature data extracted from the training libraries. This way you can make sure that you use the same data in the same order. Useful when trying out various options such as different features, |
| -Folds | <Number> | Defaults to ‘10’ for 10 fold Cross Validation. |
| -Format  or  -F | <File Format> | Format to use when reading <Feature Data File>. (-DF) defaults to PICES. <PICES | UCI | Sparse | C45 > |
| -Hierarchy | none | There will be a cross validation performed for the data specified for 4 levels of hierarchy in the class name structure. With this option the underscore (\_) characters in the class name will be used to denote levels of hierarchy.  Ex: Class Name “phytoplankton\_diatom\_rhizosoplenia” has three levels of hierarchy.  Level1: phytoplankton”  Level2: phytoplankton\_diatom  Level3: phytoplankton\_diatom\_rhizosoplenia  The level 1 cross validation will lump all classes that start with “phytoplankton\_” into a general class called “phytoplankton”. |
| -MC | <Directory> | The directory where to make a copy of miss-classified images to. If specified will create a directory tree with a subdirectory tree for each class. Within each of those subdirectories will be a directory for each class that is miss-labeld. |
| -Normalize |  | Used in conjunction with –DF. If specified then the data in the file is assumed to be already normalized. |
| -Randomize |  | If specified will randomize the order of the feature data, and stratify the classes evenly amongst the folds. |
| -ReCalc |  | Used in conjunction with –DF. If you want to use the same ordering as already exists in a feature file but want to recalculate the features, (because of Program Change). |
| -Report  or  -R | <File Name> | Name of file to print report to, if not specified will print to console. |
| -Save  or  -S | <Feature File Name> | This will save the Feature Data that is used in the Cross Validation. The data will be in the same order that it was used in. This way you can use the same data by using the –DF option with this file name. |
| -Validation  or  -V | <Validation Data File> | Use this with the “-DF” option to perform a validation.  Training will be done with the data specified with “-DF” and testing with the validation data. |

Example Usage:

=======================================================================================

CrossValidation -C “C:\TrainingApp\DataFiles\TrainingModels\ECOHAB.cfg”

-Report CrosssValidationReport.txt

-Save EchohabTestData.data

=======================================================================================

Will run a 10 fold cross validation. The report file “CrossValidationReport.txt”: will be generated and the data used will be saved in

=======================================================================================

CrossValidation -C “C:\TrainingApp\DataFiles\TrainingModels\ECOHAB.cfg”

-Report CrosssValidationReport.txt

-DF EchohabTestData.data

=======================================================================================

Will run a 10 fold cross validation. Instead of using data from the training library, data that was saved in the previous example ‘EchohabTestData.data’ will be used.

=======================================================================================

CrossValidation –Dir c:\Pices\etpC\_heirarchy

-Report ETP\_Hierarchy\_Report.txt

=======================================================================================

Will run a 10 fold cross validation. The “-dir” option tells the cross validation program to load data from the subdirectory rooted at “c:\Pices\etpC\_heirarchy”. Each directory name will be treated as a class name and each image file will be treated as an example. As a result a new configuration file called “etpC\_heirarchy.cfg” will be created and a feature file:

“c:\Pices\etpC\_heirarchy\etpC\_heirarchy.data”.

## Random Splits.

Another tool similar to CrossValidation(6.11) which will generate a Confusion Matrix indicating the performance of a specified Training Model. It is meant to run in a multiprocessor fashion such that you start up as many instances as you want, generally one for each CPU. The user will specify a “Feature Data File”, “Training Model”, “Number of random Splits”, and “Percentage to Split by.

1. Each instance of RandomSplits.exe must run in the same directory.
2. When the program starts it will create a Status File that keeps track of each instance of “RandomSplits.exe” and what each instance is doing.
3. Upon startup of the first instance of RandomSplits.exe will initialize the run by:
   1. Creating the specified number of random splits of the Feature data File. This will be done by creating an Index File that keeps track of each the different orderings of the “Feature Data File”. This way if you want to rerun using different Training Model/Configuration File you can re-run using the exact same splits. Each split will consist of Training Data and test Data split by a given percentage. All Classes will be divided by the same percentage.
   2. Initialize the Status File which will be called “RandomSplits.Status” with one entry for each random split.
4. Each instance of RandomSplit.exe will read the “RandomSplits.status” file and grab the next available entry and start processing it. When it is done it will append the results to the status file.
5. Each Split will:
   1. Calculate Bias Parameter by running a 10 fold cross validation against the training data.
   2. Build a classifier using the Training Data and test against the test data.
   3. Apply the Bias Parameters from step a.
6. When all Random Splits are done a summary reflecting all random splits will be produced.
7. Only the first invocation of “RandomSplits” will be required to provided command line parameters. The other invocations will get their parameters from the status file.

RandomSplits

-Config <Configuration File>

-DataFile <Feature data File>

[-Format <Pices | C45 | Column | DST | Pices | Sparse | UCI> ]

[-NumSplits <Number of random Splits, defaults to 30.> ]

[-Restart If specified then Status File Will be reinitialized ]

[ to show that no jobs are running. ]

[-SplitPercentage <Percentage to be used as training defaults to 70% ]

Example Usage:

=======================================================================================

randomSplits -C MegaLigray\_1 –df HRS0202.data -F Pices -NumSplits 100 –SP 80

Will start a RandomSplits session with 100 random splits where each split will be divided into 80% Training Data and 20% Test Data. The source data file will be “HRS0202.data”. The configuration file will be ”megaLibrary.cfg” which will be either in the default directory or in the “${PicesHomeDir}\DataFiles\TrainingModels” directory.

=======================================================================================

## Feature File Converter

This application “FeatureFileConverter.exe” is a command line application that can convert between different Feature Data File formats.

FeatureFileConverter

-SrcFileName <Source File Name>

-DestFileName <Destination File Name> Optional

-SrcFileFormat <C45, Column, DST, Pices, Sparse, UCI> Defaults to 'Pices’

-DestFileFormat <ARFF, C45, Column, Pices, Roberts, Sparse, UCI> Defaults to 'Pices'

-Report <Report file name> Defaults to 'cout'

-STATistics Print out feature statistics

-Features <Specific Features to extract> Defaults to 'All'

-EncodeFeatures <'Yes' | 'No'> Defaults to 'No'

-EnumerateClasses <'Yes' | 'No'> Defaults to 'No'

-Normalize Normalize data to zero mean with a plus StdDev = 1.0.

==========================================================================================

ex: FeatureFileConverter -SFN Test.data -DFN Test\_out.data -dff c45

Convert *Pices* formatted data in file 'Test.data' to *C45* formatted in “Test\_out.data”.

==========================================================================================

## Feature File Stats

FeatureFileStats.exe A command line program that will produce statistics for a specified Feature Data file by class.

## Randomize Data

This is a utility application that started out as a way to randomize the order of data in feature files. Since then it has grown to support several functions such as converting from one format to another, create Test and Validation sets and other operations. This program assists in performing experiments such as Active Learning, and Feature Selection

**RandomizeData**

-File <Data file to perform operation on>

-Action <Convert | Split | Shuffle | Test\_Validate>

[-Featutres <List of features to include>]

[-IF <Input Format>]

[-LopOff <Number of rows to keep by class>]

[-Norm Normalize Output Data]

[-OF <Output Format>]

## Merge Feature Files

MergeFeatureFiles.exe; A command line program that will merge the contents of multiple source Feature Data files into a single Feature data file. It will also allow you to convert the format at the same time; such that you can read one file in “c45” format and another in “ARFF” format and write to new file using the “PICES” format.

## Build Image Directory from a Feature File.

BuildImageDirFromFeatureFile.exe; This is a command line program that will locate images that are specified in a existing PICES Feature Data file and produce a subdirectory tree of these images such that each class will be in its own class.

## SIPPER3 Simulator.

This is a .net program that will simulate the function of the SIPPER device. Its purpose is to aid in the development and testing of the SIPPER Interface program.

# Appendixes

## System Design.

The classification system consists of several applications performing various functions such as Image Extraction, Feature Calculation, Classification, Database Management Training Library Maintenance and several reporting programs. The classification of plankton images are accomplished through machine learning techniques, specifically using a Support Vector Machine.

The code was written in c++ and c#. C# is used to provide the GUI interface while c++ is used for all back end processing. The c++ code is mostly platform independent such that it can be compiled and executed in Windows, UNIX, and Linux environments.

### Data Flow in Sipper System

Figure 36 shows the flow of data in the existing SIPPER system. It starts with raw image data in SIPPER files. The SIPPER files are images that are 10cm’s wide by kilometers in length. Discrete Plankton Images are extracted and features for each individual image are computed in the box labeled “Image/Feature Extraction. These extracted images with their corresponding feature data are then placed into “Extracted Images” pool where it awaits classification. The classification process uses a classifier, which is built from existing training examples in the “Training Library” pool, to classify the unlabeled examples in the “Extracted Images” pool. These images are then placed into the “Classified Images” pool with their corresponding assigned labels and confidence/probability values. At this point we enter a iterative process where we refine the performance of the classification process. This is done by selecting candidate images from the “Classified Images” pool that have the greatest likelihood of improving classification accuracy. These images are then added into the “Training Library” pool and used to generate a new classifier that will be used by the “Classification Process”. A more detail description of this iterative process follows below under the label “Active Learning”.



Figure Flow of data in SIPPER System.

There are several steps involved in extracting images and classifying them from raw SIPPER data. SIPPER data consists of both image and instrument (environmental) data intermixed together organized into two byte packages. A package is either for Instrument data or Image data with the most significant bit specifying which. The instrument data consists of the text lines produced by instruments such as CTD. These text lines are parsed for relevant environmental data. The image data uses a simple run length compression format organized into 4096 pixel scan lines. The instrument data is temporally close to the image data which allows us to know the exact environmental parameters at the time the Plankton particles were imaged. The extraction process will decompress the SIPPER data and produce two streams of data one consisting of scan lines and the other of instrument data.



Figure SIPPER Data Flow.

**Decode SIPPER Data**: The SIPPER raw data consists of compressed data using a simple Run Length algorithm. It is organized into two byte data records that are either Image data or instrument data. The decoder will decompress this data and create two streams of data. One consisting of 4096 pixel scan lines with 3 bit grayscale, and the other of the instrument data as produced by the original instruments such as a CTD. The instrument data will be human readable text data as produced by the Source Instrument.

**Extract Frame:** This function groups scan lines together into logical frames such that individual plankton images are not split across two frames. These frames can be between 1 and 4096 scan lines in length. This is accomplished by taking the next 4096 scan lines and starting with the last scan line work backwards until a break point consisting of three blank scan lines is detected. If no such break point is fond then the scan line with the least number of foreground pixels is used as the division point between two frames. All scan lines past the break point are added into the next frame. As part of this step a filter is applied that removes artifact lines that can be caused by the accumulation of particles that partially block the camera light source.

**Extract Individual Images:** Given a frame as input this function identifies individual plankton images by performing a connected component analysis. Two pixels are considered connected if they are both foreground and within three pixels of each other.

**Extract Features:** For each plankton image a Feature Vector is calculated. This Vector consists of 88 Features for each Plankton image. These features can be divided into several groups; Size; Moment, Morphological; Contour; Textural, and instrument data.

**Classify Images:** This function will classify unknown images into user defined classes. The user provides a library of classes which consist of user labeled plankton images. For each class the user will provide a set of examples from which the classifier will learn from. The classifier uses a Machine Learning Algorithm called a Support Vector Machine (SVM) **Invalid source specified.**. The SVM will learn from the user labeled images how to recognize the Class that the unlabeled images will be assigned to. The SVM accomplishes this by locating hyper-planes that separate the different classes from each other. For example if the user provides examples for 4 different plankton classes called "A", "B", "C", and "D"; SVM will find 6 different hyper planes which will separate each possible class combination (AB, AC, AD, BC, BD, and CD). Unknown examples/images can now be plotted in the same space as the hyper-planes and depending on what side of the surfaces they fall into the most likely class that they belong to can be derived. In effect the each example will participate in 3 different elections and the class that has the most number of votes wins. In the case of ties the class with the highest probability/confidence will be selected. The confidence/probability **Invalid source specified.** is a function of the distance from the hyper-plane an example falls.

Once images are classified they are added to the PICES Image Data Base where they can be retrieved and reports can be created from. Along with the classification a probability is assigned to indicate how likely the prediction is correct. The probability is derived from the distance from the separating hyper-planes the unknown example plots from. The greater the distance the more likely an example belongs to the class that is on the side of.

**Active Learning:** Invalid source specified. This is the process of improving the Training Library by locating examples that have a high likelihood of improving classification accuracy and having a user/expert manually classify them. This is accomplished by selecting the examples that the Classifier has the hardest time distinguishing between two possible classes. Specifically it uses the probabilities/confidence values that are assigned to each class for each Prediction. The examples that have the smallest different between the two most likely classes are selected. These images are then added to the training library where they will be used for future classifications.

**Noise Line Elimination:**

void SipperImageExtraction::EliminatePosibleLines (uint colCount[])

{

uint endCol;

bool endColFound;

uint col;

uint row;

uint startCol;

uint threshold = lastRowInFrame / 6;

if (threshold < 300)

threshold = 300;

uint x;

BuildColCount (colCount);

for (col = 1; col < (MAXLINELEN - 1); col++)

{

if (colCount[col] > threshold)

{

startCol = col;

endCol = col;

endColFound = false;

col++;

while ((col < (MAXLINELEN - 1)) && (!endColFound))

{

if (colCount[col] > threshold)

{

endCol = col;

col++;

}

else

{

endColFound = true;

}

}

int width = 1 + endCol - startCol;

if (width > MaxArtifactLineWidth)

continue;

// Now that we defined a Possible Collumn lets get rid of the stupid Line.

uchar\* rowContent = NULL;

for (row = 0; row <= lastRowUsed; row++)

{

rowContent = frame[row];

if (((rowContent[startCol - 1] == 0) || (startCol <= 1)) &&

((rowContent[endCol + 1] == 0) || (endCol >= (MAXLINELEN - 2))))

{

for (x = startCol; x <= endCol; x++)

{

if (rowContent[x] > 0)

{

rowContent[x] = 0;

pixsInRows[row]--;

}

}

}

}

}

} /\* end of for col \*/

} /\* EliminatePosibleLines \*/

During ProcessFrame

===================

if ((sipperImage->ColLeft () < 350) || (sipperImage->ColLeft () > 3700))

{

uint width = 1 + sipperImage->ColRight () - sipperImage->ColLeft ();

uint height = 1 + sipperImage->RowBot () - sipperImage->RowTop ();

if ((height > 90) && (width < 30))

noiseLine = true;

else if ((height > 300) && (width < 40))

noiseLine = true;

else if ((height > 600) && (width < 51))

noiseLine = true;

else if ((height > 1500) && (width < 58))

noiseLine = true;

}

## Feature Descriptions.

Table Feature Categories.

|  |  |  |
| --- | --- | --- |
| Category | Sub Category | Feature  Count |
| Moment Features **Invalid source specified.** | Binary | 8 |
| Intensity Weighted | 8 |
| Edge Pixels Only. | 8 |
| Morphological |  | 9 |
| Head/Tail, main access of image is found via a Eigen Vector, image rotated to align with horizontal access. | Pixel Counts of First Quarter and Last Quarter. | 2 |
| Length vs Width | 1 |
| Length | 1 |
| Width | 1 |
| Filled Area |  | 1 |
| Convex Area |  | 1 |
| Transparency | One Binary, One Weighted | 2 |
| Texture Using Fourier Transform. | One Feature for each Frequency Range from Low to High Frequency. | 5 |
| Contour Fourier | Average of 5 Frequency Domains Low to High. | 5 |
| Hybrid, 4 lowest frequencies are sampled while the rest represents ranges of frequencies. | 15 |
| Intensity Histogram | Not Including white space | 7 |
| Including White Space | 8 |
| Instrument Data | Depth | 1 |

Table Common Variables /Functions used in Feature Calculation.

|  |  |
| --- | --- |
|  | Image Height |
|  | Image Width |
|  | Intensity at x, y 0 = Background, 255 = Foreground |
|  | Center of Image |
|  | Weighted Center |
|  | Number of Foreground Pixels in Image . |
|  | Image Size in Number of Pixels |
|  | Indicates which intensity range the pixel value  is in. See Table 9.  Ex: . |
|  | Indicates that the Intensity of pixel at location falls in intensity range , 1 if true else 0. |
|  | Indicates that the Intensity of pixel at location is greater than 31, a foreground pixel, 1 if true else 0. |
|  | Histogram Feature Value for intensity range r. See Table 9. |











There are three groups of moment features, 1) moments on whole image(0-7), 2) Edge Image(8-15) , and 3) weighted(31-38). These moments features are based off work introduced by HU[[1]](#endnote-2)

Table Definition of the eight basic moment features used in the three different moment groups.

The following equations are used in Table 13 as part of the Plankton feature data description. Equation (2) computes the number of foreground pixels in image . Equation (3) returns the weighted size of the image; that is, the size is weighted by the intensity of each pixel. Equations (4) and (5) return the centroid and weighted centroid of image .

|  |  |  |
| --- | --- | --- |
|  |  | (1) |
|  |  | (2) |
|  |  | (3) |
|  |  | (4) |
|  |  | (5) |

Table 6 describes the 8 basic Moment features developed in **Invalid source specified.**. There are three different flavors of moment features implemented: binary, edge, and intensity weighted.

1) Features 0 through 7 are the binary moment features and use Equation (6) with the moment features described in Table 6.

2) Features 8 through 15 edge moment features and use Equation (6) with the moment features described in Table 6.

3) Featured 31 through 38 are the intensity weighted moment features and use Equation (7) with the moment features described in Table 7.

|  |  |  |
| --- | --- | --- |
|  |  | (6) |
|  |  | (7) |

Table Eight Basic Moment Features Used in the Three Different Moment Groups.

|  |  |
| --- | --- |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |

With the grayscale values that SIPPER 2 and SIPPER 3 produce, features that reflect the texture of the image can be computed. A 2D Fourier Transform is performed on the original image. By using the result of this transform the energy of different frequency ranges was captured by computing the average magnitude for each of 5 different frequency ranges (see Table 8).

Figure 38 shows a plankton image and its Fourier transform. The semi circle bands that are labeled R1 thru R5 indicate the boundaries of the regions. Only half the Fourier domain needs to be processed, since both halves are mirror images of each other. These five regions result in five Fourier features. The value of each feature is the average value of the magnitude of their respective region.

|  |  |
| --- | --- |
| C:\Temp\FourierTextureFeatures\ETP2008_8AA_04_32294679_2466_Orig.bmp |  |

Figure 2D Fourier Transform of Image, Frequency Ranges Indicated.

Table 7 provides descriptions of some variables and functions that are needed for equation (8). Using these equations five features are computed that represent five different frequency ranges, as listed in Table 8.

Table Texture Features Variables and Functions.

|  |  |
| --- | --- |
| Function | Description |
|  | Fourier transform of image. This will be a two dimension matrix with the same dimension same the original image. Each element in the matrix will have both a real and imaginary part. |
|  | Distance from upper left to centroid. |
|  | Indicator function that specifies weather the pixel at is in region . Return 1 if true or 0 if false. Uses Table 9 and . |
|  | Pixel count for region . |

Table Lower and Upper Frequency Bounds for Texture Features.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Region Number | |  |  | | --- | --- | | Lower Bound |  | | |  |  | | --- | --- | | Upper Bound |  | |
| 1 | 0 |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  | 1 |

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

Equation (9) computes the fraction of image pixels that belong to a given range. It is used by the two groups of intensity histogram features. The first group, features 63 through 69, is computed from the original image, while the second group, features 74 through 82, is computed after a fill-hole operation is performed on the original image.

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

Table Intensity Regions.

|  |  |
| --- | --- |
| Region | Intensity Range |
| Background | 0 - 31 |
| 1 | 32 - 63 |
| 2 | 64 - 95 |
| 3 | 96 - 127 |
| 4 | 128 – 159 |
| 5 | 160 – 191 |
| 6 | 192 – 223 |
| 7 | 224 – 255 |

Contour features based off Fourier descriptors were implemented. Fourier descriptors were derived by performing a Fourier transform on a one dimensional array of data that represents the contour of the image, where the real and imaginary components come from the locations of the edge pixels. When performing a Fourier transform on an array that represents the edge/contour of an image, the frequencies captured in the resultant array will reflect the deviations from a circle. There are two types of contour features implemented: 1) averaging by frequency region, and 2) a combination of region averaging and sampling referred to as hybrid.

1) A Fourier transform is performed on the entire contour of the image. The result of the transform is used to generate 5 contour features with each one representing a range of frequencies. This is done by computing the average value of the magnitudes for each range, (see Figure 39). This is similar as to the way the Texture Features were computed. In this case instead of bounding the regions with semi-circles around the center of the image, the region is derived by determining the distance from the center of the array. Table 11 shows the size of the frequency ranges as a fraction of 1. Equation (10) computes the averaging contour feature for the specified region using functions described in Table 11.

2) Hybrid is a mix of averaging and sampling. The lowest frequencies are sampled and the higher frequencies are averaged. The idea here is that the lowest individual frequencies capture the greatest amount of information, while individual higher frequencies are not as significant but taken as an average over a domain can contribute to classification accuracy. Table 12 gives a summary of the 16 features computed in this section.

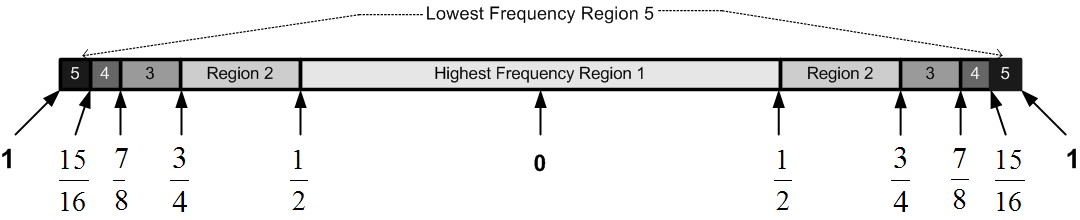


Figure 39 Contour Frequency Domain.

Table Upper and Lower Contour Frequency Ranges.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Region  Number | |  |  | | --- | --- | | Lower  Bound |  | | |  |  | | --- | --- | | Upper  Bound |  | |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

Table Contour Variables and Functions.

|  |  |
| --- | --- |
| Variable | Description |
|  | Length of edge in pixels |
|  | Center position |
|  | Magnitude of complex number(amplitude) at position. |
|  | Indicator function, specifies weather position is in region .  If  then 1 else 0. |
|  | Number of pixels in region |
|  | Contour Feature Value for region . |

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

Table Hybrid Contour Features.

|  |  |  |
| --- | --- | --- |
| 0 | 1 Hz Left | First Bucket in resultant Fourier transform |
| 1 | 2 Hz Left | Second Bucket in resultant Fourier transform. |
| 2 | 3 Hz Left |  |
| 3 | 4 Hz Left |  |
| 4 | 13/16 – 4 Hz | Avg. of amplitudes in left buckets that range from 13/16th to 4hz from center |
| 5 | 12/16 – 13/16 | Avg. of amplitudes in left buckets that range from 12/16th to 13/16th from center |
| 6 | 10/16 – 12/16 | Avg. of amplitudes in left buckets that range from 10/16th to 12/16th from center |
| 7 | Center – 10/16 | Left |
| 8 | Center – 10/16 | Right |
| 9 | 10/16 – 12/16 | Avg. of amplitudes in right buckets that range from 10/16th to 12/16th from center |
| 10 | 12/16 – 13/16 | Avg. of amplitudes in right buckets that range from 12/16th to 13/16th from center |
| 11 | 13/16 – 4 Hz | Avg. of amplitudes in right buckets that range from 13/16th to 4hz from center |
| 12 | 4 Hz Right |  |
| 13 | 3 Hz Right |  |
| 14 | 2 Hz Right |  |
| 15 | 1 Hz Right | Last Bucket in resultant Fourier transform |

Table List of Plankton Features.

|  |  |  |
| --- | --- | --- |
| Feature  Num | Description | |
| 0 – 7 | Moment Features from Table 6 using equation (6). | |
| 8 – 15 | Edge Moments   1. Image is reduced to just edge pixels. 2. New center  is calculated. | |
| 16 |  | |
| 17 |  | |
| 18 |  | |
| 19 |  | |
| 20 |  | |
| 21 |  | |
| 22 |  | |
| 23 |  | |
| 24 |  | |
| 25,26 | a. Create covariance matrix of image .  b. Calculate 1st and 2nd Eigen vectors of .  c.  d. Determine orientation of image,  e. Using orientation , rotate Image so that it lies horizontal.  f. ,  g. , Helps to determine if organism has head. | |
| 27 |  | |
| 28 |  | |
| 29 |  | |
| 30-37 | Moment Equations from Table 6 using equation (7). | |
| 38-42 | , , , and | |
| 43-57 | Hybrid contour as described in Table 12. | |
| 58-62 | Averaging contour as described in  Table 10 and equation (10). | |
| 63-69 | Intensity Histogram Field [0-6] | |
| 70 | Height / Width, Using information used to calculate the eigen ratio  a. Image is rotated so that its longest dimension run parallel to bottom.  b. Tight bounding box is drawn.  c. The shortest dimension is considered the Height while the longest is the Width. | |
| 71 | Height | |
| 72 | Width | |
| 73 | Hole filled area. | |
| 74 | FlowRate1 From SIPPER’s Flow Meter. | |
| 75 | FlowRate2 From SIPPER’s Flow Meter. | |
| 76-83 | Intensity Histogram including whitespace. | |
| 84 | Depth from | CTD Derived Fields **Invalid source specified.** |
| 85 | Salinity |
| 86 | Oxygen |
| 87 | Fluorescence |

## SIPPER3 Raw Data Layout

The following describes the layout of the RAW SIPPER data file as produced by the SIPPER 3 device. This information was derived from e-mail communications with William Flannery. SIPPER 3 produces a continuous stream of 16 bit records that read from most significant bit (MSB) to least significant bit (LSB). Table 16 gives a detailed description of each bit. Each 16 bit record contains either image data or instrument data, as specified by bit 15. The two types of records are processed by separate decoding functions.

There are three basic types of image data records: Gray Scale, White-Run-Length, and binary. The first two types are the most common. The third type, binary, only occurs when SIPPERS’ internal buffer is getting full and needs to write the data to disk before an overrun occurs.

1) Gray Scale records provide four grayscale 3 bit pixels that range from 0 to 7 where 0 represents white (background) and 7 represents black. These values are scaled to 8 bit range, as indicated in Table 14, to aid in compatibility with future versions of SIPPER, where 8 bit level grayscale is envisioned. When data is stored in image files such as BMP images, the values are complemented such that 255 = 0, and 0 = 255.

Table SIPPER 3 Grayscale Decoding Values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 Bit  Value | 8 Bit  Scaled  Value |  | 3 Bit  Value | 8 Bit  Scaled  Value |
| 0 | 0 | 4 | 146 |
| 1 | 36 | 5 | 182 |
| 2 | 73 | 6 | 219 |
| 3 | 109 | 7 | 255 |

2) White-Run-Length records are an implementation of a simple run-length compression algorithm. The majority of SIPPER data is white background. This record will specify the number of 4 pixel packages that contain white that occur in a row. The count is specified in bits 11 through 0 and is multiplied by 4 to get the number of white background pixels that have occurred.

3) Binary image data. This format is meant to help prevent buffer overflows. Since only white or black are being recorded SIPPER can write four times more data in a given amount of time. The downside is that texture information is lost. In practice this situation rarely occurs; the most common cause is when bubbles pass through the sampling tube, which can occur when SIPPER is very near the surface, such as when first being deployed. In this case white pixels are mapped to 0 and black pixels to 255.

Instrument Data has two different formats, text and binary data. In practice only the text variation is used. Each 16 bit record contains a 6 bit sensor number. Table 15 contains a list of sensor numbers that are currently in use.

Table SIPPER File Sensor Number Descriptions.

|  |  |  |
| --- | --- | --- |
| Sensor  Number | Name | Description |
| 6 | User Message | This is a user provided description provided via the SIPPER interface. It is written on the SIPPER disk at the beginning of the SIPPER file. The Disk Manager software, which is used to offload SIPPER Files, reformats this data into the 16 bit records as described in Table 16. |
| 9 | GPS Data | As of this time has not been implemented. GPS data is currently being imported into the PICES database from text files provided by hosting research vessels. |
| 10 | Flow Rate | This instrument consists of both text and binary data. The text indicates the half turns of the flow meter where there are 98 turns per meter. The binary data indicates flow rate in meters per sec. |
| 16 | Serial Port 0 | CTD , note CTD can have up to 4 external instruments, such as O2 senor, who’s data will be included with CTD data |
| 17 | Serial Port 1 | Pitch and Roll Sensor. This is text only data. Each line is separated by a line feed character.  Ex: "R -1.15 P 16.18” |
| 18 | Serial Port 2 | Battery Sensor. Provides voltage levels and status of SIPPERS 4 batteries.  Ex: "1, 25.55, 26.61, 26.14, 25.79, LLLL"  Active battery followed by 4 voltage readings, followed by Live/Dead status of 4 batteries. Batteries are labeled 1,2,3,4. |
| 19 | Serial Port 3 | Unused. |

Table Data Payload Table

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit# | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Bit  Name | Image | EOL /  ASCII | RAW | Gray | Data | | | | | | | | | | | |
| 8 | 1 | 0 | 0 | 0 | Compressed, count of blocks of 4 white pixels.  Ex: 0x312 = 786 = (786 \* 4) = 3144 white pixels. | | | | | | | | | | | |
| 9 | 1 | 0 | 0 | 1 |
| A | 1 | 0 | 1 | 0 | Black and white, 8 binary pixels. | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| B | 1 | 0 | 1 | 1 | Gray Level  pixel 1 | | | Gray Level  pixel 2 | | | Gray Level  pixel 3 | | | Gray Level  pixel 4 | | |
| C | 1 | 1 | 0 | 0 | Compressed, count of blocks of 4 white pixels. Followed by end-of-line. | | | | | | | | | | | |
| D | 1 | 1 | 0 | 1 |
| E | 1 | 1 | 1 | 0 | End Of Line encountered, so there are 4, 8, or 12 pixels of Black  and White stored incrementing left to right as above. The program will have to count to know which pixels are valid. | | | | | | | | | | | |
| F | 1 | 1 | 1 | 1 | 4 gray-scale pixels, as above, followed by End of Line. | | | | | | | | | | | |
|  | 0 | 0 | Sensor number | | | | | | Sensor data | | | | | | | |
|  | 0 | 1 | Sensor number | | | | | | Sensor-related text | | | | | | | |
|  |  | Some sensor numbers are defined. See Table 15. | | | | | | | | | | | | | | |

## Feature Data File Formats.

The command line programs support different formats for the Feature Data Files. The default one is called Pices which is the native format developed for the PICES system.

|  |  |  |
| --- | --- | --- |
| Format Name | Read Supported | Write Supported |
| Pices | Yes | Yes |
| C45 | Yes | Yes |
| Sparse | Yes | Yes |
| ARFF | No | Yes |
| Col | Yes | Yes |
| DST | Yes | No |
| UCI | Yes | Yes |

## Software Architecture.

The PICES system is written in a mix of c++ and c#. All the GUI software is in c# and runs under the windows platform while all the back end processing is written in c++ and is compatible with both the windows, linux, and solaris platforms.

Table Libraries used by PICES Software

|  |  |  |
| --- | --- | --- |
| Outside Source Code and libraries. | | |
| Fftw | Fastest Fourier Transform in the West  Used to aid in the generation of texture and contour features. | <http://www.fftw.org/> |
| Zlib123 | Zlib Data Compression Library  Used to compress images before storing in MySql database. | <http://www.winimage.com/zLibDll/index.html> |
| Unmanaged c++ code that is compatible with Windows, Linuix, and Solaris. | | |
| BaseLibrary | Non Application Specific routines that support functions for Image Processing, String Management, Token Parsing and other useful functions | |
| SipperInstruments | This code is specific to the SIPPER device; it has routines that can read raw SIPPER files from versions 1, 2, and 3. It can parse out SIPPER Instrumentation data such as CTD data. It is also aware ofg the PICES software installation; it is aware of where all client data files are located via the class “SipperVariables”. | |
| JobManager | A general Job Management Library used to help applications that are to work in a multiprocessing environment. This is a relatively new library which is only used by RandomSplits at this time. Applications like FeatureSelection will eventually be modified to make use of this library. | |
| PicesLibrary | This library is PICES Aware; it has all the Machine Learning and Feature Extraction code. At some point in the future it will need to be split into at least three different libraries. One that supports the general Machine Learning World, one that supports the feature extraction that is specific to PICES and the third that supports the Database interface to MySQL. | |
| libSVM | <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>  This consists of a single module that is included in the PicesLibrary. It has been heavily modified. | |
| Managed c++ and c# code meant to run under Windows .net world. | | |
| PicesInterface | A managed c++ library that acts as an interface from the .net world to the unmanaged code in the “SipperInstruments”, “BaseLibrary”, and “PicesLibrary” world. | |
| SipperFile | This is a managed c# library that provides assorted routines for all the .net GUI applications. It also includes routines for accessing raw SIPPER files, this is duplicated code that can also be found in “SipperInstruments” | |
| SipperDiskUtilities | A c# library that provides routines that allow us to access SIPPER disk drives. The Fire-wire drives that the SIPPER device writes to uses a proprietary format. These routines provide functions that allow us to read the SIPPER disk directory, read individual SIPPER files, and reinitialize a SIPPER drive for preparation for a deployment. This library is primarily used by the SipperDiskManager application. | |

## Database Description.

We use a MySQL database server. This server can be hosted on any machine that MySQL supports. The PICES Database consists of 14 tables that keep track of things such as Cruise, Stations, Individual Deployments, Sipper Files, Individual Plankton Images, Groups of images, and other auxiliary support items. Table 18 gives a quick description of each table. Most queries and updates are done through stored procedures. There are a few exceptions that will be eliminated in future releases of PICES. All stored procedures are called through a single module called “DataBase.cpp” in PicesLibrary. This should allow easy port to other databases systems, such as Microsoft SQL, in future releases.

### Table Descriptions:

Table List of Tables in PICES Database.

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Description** | **Primary Key** |
| Classes | There will be one entry for each Class of plankton that you wish to keep track of plus additional entries that help support a hierarchy structure. Example There will be an entry for “Crustacean\_Copepod” that will represent all Copepod classes. In addition to this entry you will have several other entries representing different types of copepods such as “copepod\_calanoid”, “copepod\_copilia”, and “copepod\_nauplii”. See Figure 24 on page 28. | Class ID |
| Cruises | There will be a single entry for each cruise that data is collected for. This will allow the PICES system to keep track of images by cruise. Within each cruise we will be keeping track of Stations, and within each Station there will be Deployments. His table is maintained via a Maintenance function on PicesCommander (see paragraph 6.3.3.2 on page 29 ). | Cruise Name |
| CtdExternalSensors | There will be one entry for each instrument that can be attached to the CTD. This will be used by the SIPPER File maintenance function to validate configurations entered by the user. | CTDExternalCode |
| Deployments | This table is related to the Stations table. There will be one entry for each deployment of Sipper. There can be multiple Deployment entries for each Stations table entry. For each Deployments table entry there can be multiple entries in the SipperFiles table. (see paragraph 6.3.3.2 on page 29). | CruiseName,  StationName,  DeploymentNum |
| FeatureData | This is where feature data for each Image is stored. There can only be one entry for each Image in the Images table. It is not required that there be an entry in this table though. The main source of entries in this table is the ImageExtraction program. As it extracts images and computes features it will also update this table. PicesCommander can also add images to this table if has to compute features for a image. | ImageId |
| ImageGroup | The PICES system can also keep track of groups of images. This table is where the Group names are stored. Typically a group can be all the images that make up a Training Library, or a group of Randomly Harvested Images(see paragraph 6.3.6.2 on page 33). | GroupeId |
| ImageGroupEntries | This table ties images to groups. An image can belong to multiple groups so for each group that an image belongs to there will be in entry in this table. | GroupeId,  ImageId |
| ImageQueryLog |  |  |
| Images | There is a single entry for each Image that PICES is keeping track of. In this table will be the SIPPER File that the image came from, the location in that SPPER file (Scan Line, Column, and Byte Offset), the Depth, Size, and Classification. There will be three classification fields. The two most likely classes as assigned by the classifier with their related probabilities and the Validated class. The Validated class is the class that the expert user has assigned to the particular image. Also a Thumbnail version of the image will be saved in this table. | Image Id |
| ImagesFullSize | Used to store the full size version of the plankton image. If the image is larger than the Thumbnail stored in the Images table then the full size version will be stored in this table. | Image Id. |
| InstrumentData | Instrument data that is collected by SIPPER will be stored in this table. Each row will contain all instrument readings that are recorded. This includes CTD data (Temperature, Salinity, Pleasure, CTD external instruments), Flow Rates, Turbidity, Pitch and Roll, and Battery Status. There will be a row recorder ever 4096 scan lines which represents ~ 10 center meters. So to get the instrument data for a particular image you would index into this table by SIPPER File and Scan Line. | Sipper File Id,  Scan Line |
| Instruments |  | Instrument Code |
| SipperFiles | For each SIPPER File there will be one entry in this table. Each entry will record what instruments were attached to SIPPER, Camera Scan Rate, SIPPER File Size (Bytes and Scan Lines), GPS location, and what Cruise, Station, Deployment that this SIPPER file is from. | SipperFileId |
| Stations | For each Station that SIPPER is deployed at an entry will be made in this table. For each one of these entries there can be multiple entries in the Deployments table. (see paragraph 6.3.3.2 on page 29). | Cruise Name,  Station Name |

### Important Fields.

**ClassId**: Every Class that the PICES Database will be keeping track of will be a sequentially assigned unique id. This way when other Tables need to reference a specific class they will refer to the ClassId not the class name. An example of this will be the Images table which will have a entry for every image that the PICES database will track. It has three fields that reference the two most likely predicted classes and the user validated class. This will allow the user to change class names without having to update all the entries in the Images table.

**ImageId**: Every image that is stored in the PICES Database will have a sequentially assigned unique id called ImageId. There will be an entry in the Images table of the database for each image. ImageId is the number that will allow us to tie that entry to other tables such as the FeaturesData and ImageGroupEntries.

**SipperFileId**: A sequentially assigned unique id that will be assigned to every SipperFile that the PICES database will be tracking. This id will help other tables tie there entries back to the SipperFile from where they originated from. For example every entry in the Images table points to the row in the SipperFile table with this id.

### Useful Scripts

## Future Work

### Training Model Maintenance Application.

### Support Vector Locator.

This would be a function that when given a specified Training Model it will provide a list of all images that became support vectors; and for what binary SVM classifiers they became support vectors for.

### Reasonable Training Image Locator.

For a given Training Model and a given image this function will locate the Support Vectors that are most likely responsible for its classification. This would be used by users when they see that an image was mis-classified; especially one that was very badly misclassified. They would get to see what Support Vectors had the greatest impact on its classification; the idea is that this would make it easier for the user to locate invalid data in the Training Library.

### Full GUI Based Installation Program.

## Publications

### SIPPER Journal Papers

Kurt A. Kramer, Lawrence O. Hall, 2 Fellow, IEEE, Dmitry B. Goldgof, Andrew Remsen, and Tong Luo, "Fast Support Vector Machines for Continuous Data", *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Pending: pages, 2009.([PDF](http://figment.csee.usf.edu/%7Ekkramer/sipper/papers/BitReduction.pdf)) ( [Data Sets](http://figment.csee.usf.edu/%7Ekkramer/sipper/papers/BitReduction/BitReductionPaper.html))

T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, T. Hopkins, "Active Learning to Recognize Multiple Types of Plankton", Journal of Machine Learning Research, JMLR 6(Apr): 589-613, 2005.([**PDF**](http://figment.csee.usf.edu/~kkramer/sipper/papers/jmlr05.pdf)) ( [**Data Sets**](http://figment.csee.usf.edu/~kkramer/sipper/papers/JMLR/jmlrPaper.html)) .

T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, T. Hopkins, "Recognizing Plankton Images from the Shadow Image Particle Profiling Evaluation Recorder", IEEE Transactions in System Man and Cybernetics, B, 34(4), pp. 1753-1762, 2004. ([**PDF**](http://figment.csee.usf.edu/~kkramer/sipper/papers/planktonVIM/planktonvim_Finale.pdf)).

A. Remsen, S. Samson, T. Hopkins, "What you see is not what you catch: A comparison of concurrently collected net, optical plankton counter (OPC), and Shadowed Image Particle Profiling Evaluation Recorder (SIPPER) data from the northeast Gulf of Mexico.

S. Samson, T. Hopkins, A. Remsen, L. Langebrake, T. Sutton, J. Patten, "A system for high resolution zooplankton imaging" IEEE Journal of Oceanic Engineering, 26(4), pp. 671-676, 2001.

### SIPPER Conference Publications

T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Sampson, A. Remsen, T. Hopkins, "Active Learning to Recognize Multiple Types of Plankton", International Conference on Pattern Recognition (ICPR), Cambridge, UK, August 2004. ([**PDF**](http://figment.csee.usf.edu/~kkramer/sipper/papers/icpr04.pdf)) .

T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Sampson, A. Remsen, T. Hopkins, "Learning to Recognize Plankton", IEEE International Conference on Systems, Man, and Cybernetics, Washington, D.C., pp. 888-893, October 2003.

### Other References

[] Hi, M.K., Visual Pattern Recognition by Moment Invariants, IRE Trans. Information Theory, 1962. pp 179-187.

[] Ron Kohavi and George H. John, Wrappers for Feature Subset Selection, Artificial Intelligence archive, December 1997, vol 97, pages 273-324.

[] M.K. Hu Visual pattern recognition by moment invariants. IEE Trans. Information Theory, pages 179-187, February 1962.

[] Nello Cristianini and John Share-Taylor, An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, 2000

[] Chih Chung and Chih Jen Lin, A Library for Support Vector Machines, <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.

## Change Log

|  |  |
| --- | --- |
| 2015-July-2 | Increased width of “DataBaseUserName” field in logentries.  ALTER TABLE `pices\_new`.`logentries`  CHANGE COLUMN  `DataBaseUserName` `DataBaseUserName`  VARCHAR(64) NOT NULL  DEFAULT ''  COMMENT 'UserName used to connect to the database with.' ; |
|  |  |
|  |  |
|  |  |
|  |  |

2015-July-2

Increased width of “DataBaseUserName” field in logentries.

ALTER TABLE `pices\_new`.`logentries`

CHANGE COLUMN `DataBaseUserName` `DataBaseUserName` VARCHAR(64) NOT NULL DEFAULT '' COMMENT 'UserName used to connect to the database with.' ;

# Refernces

**There are no sources in the current document.**

1. [↑](#endnote-ref-2)