Appendix 4. Sample batch scripts

Contains two scripts by Ken Stuart: PanoImageSorter.pl and PTMacPanoImageSorter.pl (for MacPerl).

PanoImageSorter.pl

```
#!/usr/bin/perl
# This script takes sets of pano image files and sorts them into
# folders and bracketed sets, saving a lot of time compared to
# manually creating new folders, naming them, and then moving
# files from the original to the sets. The specified folder must
# contain a complete sets of files, i.e., if there are 10 nodes
# and each node consists of 8 shots, each of which has 3 bracketed
# images, then the folder must contain 240 images exactly.

use strict;

# Path and name of folder with pano images, e.g.
# /Users/jdoe/Desktop/Panos/originals
my $StartingFolder = "/Users/kenstuart/Desktop/originals";

# Path and name of folder to store sorted images.
my $EndingFolder = "/Users/kenstuart/Desktop/project";

# Number of images per pano, usually 4-8, including top and
# bottom.
my $NumberOfImagesPerPano = 8;

# Number of frames per shot, usually 3 if bracketing, or 1 if
# no bracketing.
my $NumberOfBrackets = 3;

# Names, in order, that you want to call bracketed sets, e.g.,
# normal, dark, light. You should have the same number of terms
# as number of brackets; if no brackets, you may wish to have a
# generic term.
my @BracketNames = ("Normal", "Dark", "Light");

# Number of nodes shot.
my $NumberOfNodes = 17;

# Enter a set of names for the folders that will contain each
# node, e.g., apse, nave, northaisle, southaisle. If you leave
# the list of node names blank, the script will assign them names
```

```
# in the format "Node_n" where n is the number of the node.
# my @NodeNames = ("A", "B", "C");
my @NodeNames = ();
my $number = 0;
if ($#NodeNames == -1) {
   for (my $i=1; $i<=$NumberOfNodes; $i++) {
      if ($i < 10) {$number = "0".$i;} else {$number = $i;}
      push(@NodeNames, "Node_".$number);
   }
}

# Enter file extension used to in image filenames, usually either
# .jpg or .tif.
my $ImageFileNameExt = ".jpg";

# Check starting folder.
unless (-e $StartingFolder) {print "Could not find starting folder
$StartingFolder. Please check its name in this script on and on
the machine. Names may be case-sensitive, and should not contain
certain special characters such as spaces, slashes or colons.
Cannot continue until problem is resolved.\n\n"; exit;}

# Load image files into a list and sort it.
opendir (DIRLIST, $StartingFolder) || die print "Cannot open
directory $@.";
my     @ImageFiles     =     grep     {/$ImageFileNameExt$/i}
readdir(DIRLIST);
@ImageFiles = sort(@ImageFiles);
close (DIRLIST);

# Confirm correct number of images in directory.
my $neededImages = $NumberOfImagesPerPano * $NumberOfBrackets *
$NumberOfNodes;
my $imageCount = $#ImageFiles+1;
print "Number of images needed: $neededImages; number found in
$StartingFolder: $imageCount\n";
if ($neededImages != $imageCount) {
   print "Cannot continue. Please check numbers and image set.\
n\n";exit;
}

# Make destination folder.
unless (-e $EndingFolder) {mkdir $EndingFolder;}

# Make new folders for each node and each bracket and move images
# to them.
```

```
my $count = 0;
foreach my $nodeName (@NodeNames) {
   for (my $i=0; $i<$NumberOfImagesPerPano; $i++) {
      unless     (-e     "$EndingFolder/$nodeName")     {mkdir
      "$EndingFolder/$nodeName";}
         foreach my $bracketName (@BracketNames) {
            unless  (-e  "$EndingFolder/$nodeName/$bracketName")
            {mkdir "$EndingFolder/$nodeName/$bracketName";}
         print "Moved $StartingFolder/@ImageFiles[$count] to $En
dingFolder/$nodeName/$bracketName/@ImageFiles[$count]\n";
         rename ("$StartingFolder/@ImageFiles[$count]", "$Endin
gFolder/$nodeName/$bracketName/@ImageFiles[$count]");
         $count++;
      }
   }
}
```

PTMacPanoImageSorter.pl

```
#!/usr/bin/perl
# This script takes sets of pano image files and sorts them into
# folders and bracketed sets, saving a lot of time compared to
# manually creating new folders, naming them, and then moving
# files from the original to the sets. The specified folder must
# contain a complete sets of files, i.e., if there are 10 nodes
# and each node consists of 8 shots, each of which has 3 bracketed
# images, then the folder must contain 240 images exactly.

# This script also loads a PTMac template file, modifies it to use
# the images files, and saves it for use in PTBatch.
use strict;
use MacPerl;
# Used to set file type and creator for output template file, so
# that PTBatch recognizes it.

# Path and name of folder with pano images, e.g. /Users/jdoe/
# Desktop/Panos/originals
my $StartingFolder = "/Users/kenstuart/Desktop/Carter";

# Path and name of folder to store sorted images.
my $EndingFolder = "/Users/kenstuart/Desktop/Carter";

# Path and name of PTMac template file.
my  $PTMacTemplateFile  =  "/Users/kenstuart/Desktop/template.
ptmac";
```

```
# Number of images per pano, usually 4-8, including top and
# bottom.
my $NumberOfImagesPerPano = 9;

# Number of frames per shot, usually 3 if bracketing, or 1 if
# no bracketing.
my $NumberOfBrackets = 1;

# Names, in order, that you want to call bracketed sets, e.g.,
# normal, dark, light. You should have the same number of terms
# as number of brackets; if no brackets, you may wish to have a
# generic term.
my @BracketNames = ("Auto");

# Number of nodes shot.
my $NumberOfNodes = 17;

# Enter a set of names for the folders that will contain each
# node, e.g., apse, nave, northaisle, southaisle. If you leave
# the list of node names blank, the script will assign them names
# in the format "Node_n" where n is the number of the node.
# my @NodeNames = ("A", "B", "C");
my @NodeNames = ();
my $number = 0;
if ($#NodeNames == -1) {
   for (my $i=1; $i<=$NumberOfNodes; $i++) {
      if ($i < 10) {$number = "0".$i;} else {$number = $i;}
      push(@NodeNames, "Node_".$number);
   }
}

# Enter file extension used to in image filenames, usually either
# .jpg or .tif.
my $ImageFileNameExt = ".tif";

# Load PTMac template file into memory.
open(TEMPLATE, $PTMacTemplateFile) || die ("Could not find starting
folder $PTMacTemplateFile. Please check its name in this script
on and on the machine. Names may be case-sensitive, and should
not contain certain special characters such as spaces, slashes
or colons. Cannot continue until problem is resolved.\n\n");
my @template = <TEMPLATE>;
close(TEMPLATE);
my $template = "";
foreach my $line (@template) {$template .= $line;}
```

```
# Check starting folder.
unless (-e $StartingFolder) {print "Could not find starting folder
$StartingFolder. Please check its name in this script on and on
the machine. Names may be case-sensitive, and should not contain
certain special characters such as spaces, slashes or colons.
Cannot continue until problem is resolved.\n\n"; exit;}

# Load image files into a list and sort it.
opendir (DIRLIST, $StartingFolder) || die print "Cannot open
directory $@.";
my     @ImageFiles     =     grep     {/$ImageFileNameExt$/i}
readdir(DIRLIST);
@ImageFiles = sort(@ImageFiles);
close (DIRLIST);

# Confirm correct number of images in directory.
my $neededImages = $NumberOfImagesPerPano * $NumberOfBrackets *
$NumberOfNodes;
my $imageCount = $#ImageFiles+1;
print "Number of images needed: $neededImages; number found in
$StartingFolder: $imageCount\n";
   if ($neededImages != $imageCount) {
   print "Cannot continue. Please check numbers and image set.\
n\n";
exit;
}

# Make destination folder.
unless (-e $EndingFolder) {mkdir $EndingFolder;}

# Make new folders for each node and each bracket and move images
#  to  them.  Also,  create  PTMac  template  file  for  each  node/
# bracket.
my $count = 0;
my %templates;
my $templateFileName = "";
my $templateStr = "";
foreach my $nodeName (@NodeNames) {
   for (my $i=0; $i<$NumberOfImagesPerPano; $i++) {
      unless     (-e     "$EndingFolder/$nodeName")     {mkdir
"$EndingFolder/$nodeName";}
      foreach my $bracketName (@BracketNames) {
         unless (-e "$EndingFolder/$nodeName/$bracketName") {
            mkdir "$EndingFolder/$nodeName/$bracketName";
            $templates{$bracketName} = $template;
         }
```

```
        print "Moved $StartingFolder/@ImageFiles[$count] to
$EndingFolder/$nodeName/$bracketName/@ImageFiles[$count]\n";
        rename ("$StartingFolder/@ImageFiles[$count]", "$Endin
gFolder/$nodeName/$bracketName/@ImageFiles[$count]");
        $templateStr = "$EndingFolder/$nodeName/$bracketName/@
ImageFiles[$count]";
        $templateStr =~ s#/#:#g;
        $templates{$bracketName}=~s/path:filename/$templateStr/
is;
        $count++;
      }
   }
   while((my $key, my $value) = each(%templates)) {
      # Save template file.
      $templateFileName = "$EndingFolder/$nodeName"."_"."$key.
ptmac";
      open(TEMPLATE, ">$templateFileName");
      print TEMPLATE $value;
      close (TEMPLATE);
      MacPerl::SetFileInfo("PTMC", "TEXT", $templateFileName);
   }
}
```