Pipeline Programming Basics

This Technical Guide introduces programming basics for setting up and executing an image-processing pipeline in a geospatial script. The illustration diagrams an example: a script to resample an image in a Project File to match a reference image with the output directed to a Project File. (Code excerpts and comments for this script, PipelineResampleToMatch.sml, are on the reverse.) The pipeline in this script includes two sources (input and reference images), one filter (resampling), and one target (see the Technical Guide entitled *Geospatial Scripting: Pipeline Image Processing* for definitions). The example also demonstrates that, although the pipeline architecture is designed to encapsulate image data, properties, and operations, a script can obtain properties from pipeline stages to check for errors and compute values to be used later in the pipeline.

Pipeline stages in the Geospatial Scripting Language (SML) are *classes* in the IMAGE_PIPELINE group. The pipeline classes used in this sample script are located there, namely:

- IMAGE_PIPELINE_SOURCE_RVC: an image source created from a raster object in a MicroImages Project File
- IMAGE_PIPELINE_FILTER_RESAMPLE: filter to resample and reproject an image
- IMAGE_PIPELINE_TARGET_RVC: pipeline target that creates a raster object in a Project File
- IMAGE_PIPELINE_GEOREFERENCE: support class to access georeference properties obtained from a pipeline stage

The complete list (with documentation) of available pipeline classes as well as other SML classes and functions can be found in the Script Reference window opened from the SML Editor.

Constructing Pipeline Stages

Classes in SML have *members* (properties that can be read or in some cases changed by your script) and *methods* (predefined functions that the script can call to operate on class data). Pipeline classes can also have special methods called *constructors* for declaring a unique instance of the class and defining its parameters. The constructors for source stages, for example, require that you specify the image to be assigned to that particular source. Source stages can refer either to raster objects in Project Files or to external image files such as GeoTIFF. In this sample script the SOURCE_RVC class is used and each source instance is constructed using an instance of class RVC_OBJITEM (in the RVC SYSTEM class group) that refers to the required raster object in a Project File. The script uses a function called DlgGetObject() in the Popup Dialog function group to provide a dialog allowing the user to choose the required raster and populate an instance of RVC_OBJITEM.

Constructors for filter stages require that you specify at least one input stage and may require additional parameters. The FILTER_RESAMPLE constructor used here, for example, requires a parameter specifying the resampling method to use. Filter stages can also have different versions of their constructors with different required parameters. There are several versions of the FILTER_RESAMPLE constructor; the one used here specifies a reference image to match in coordinate reference system and cell size, whereas another version omits the reference image and requires an output coordinate reference system and cell size instead.

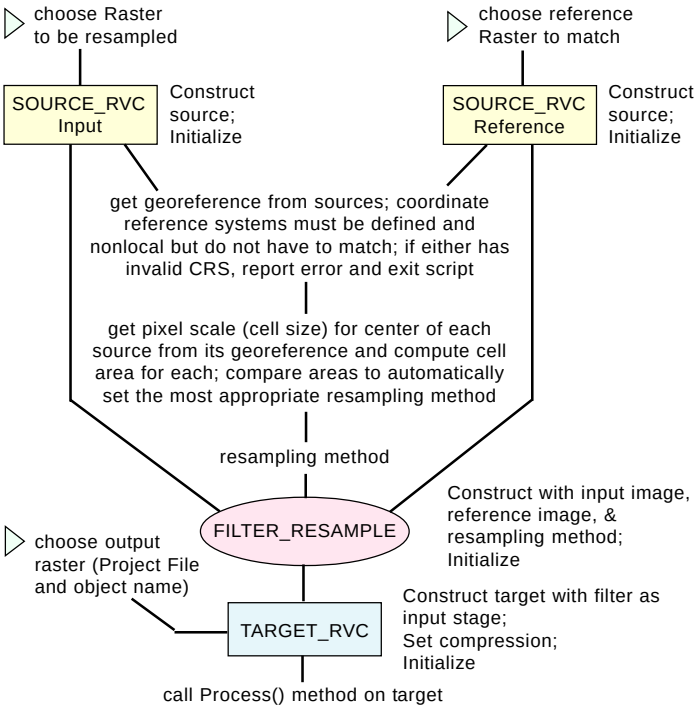
Constructors for a target stage require that you specify the input filter and identify the output image. For a TARGET_RVC, you pass the constructor an RVC_OBJITEM corresponding to the new output raster object. (Sources and targets for external image types, such as TIFF, are constructed using a FILEPATH class instance, as illustrated in sample scripts PipelineContrastCompositeToTIFF.sml and PipelineNDVIfromTIFF.sml.) Once it is constructed, you can set a compression option for a TARGET_RVC instance using the SetCompression() class method.

Initializing Pipeline Stages

Each pipeline stage class has an Initialize() method that must be called before the pipeline can use that stage. This method checks that all of the specified input stages are properly defined and returns an error code (a negative number) if any do not. As illustrated by the sample script, for debugging purposes a pipeline script should check this error value and return an error to the console or to a popup message dialog to report any errors. This sample script also demonstrates a higher level of error checking: it gets the georeference definition from both input and reference sources to check that each has a valid (though potentially different) coordinate reference system that can then be properly used in the resampling filter.

Processing the Pipeline

Once all of the pipeline stages are defined and initialized, the script initiates processing by calling the Process() class method on the target stage. This method pulls the images through all of the steps of the pipeline and creates the final result image in the designated location.



Annotated diagram of pipeline to resample an image to match a reference image (see PipelineResampleToMatch.sml on reverse).

Many sample scripts have been prepared to illustrate how you might use the features of the TNT products' scripting language for scripts and queries. These scripts can be downloaded from www.microimages.com/downloads/scripts.htm.

Pipeline Script to Resample/Reproject Image to Match a Reference Image (PipelineResampleToMatch.sml)

```
numeric err;                                    error code returned

proc ReportError(numeric linenum, numeric err) {        define error
   printf("FAILED -line: %d, error: %d\n", linenum, err);   checking
   PopupError(err);                                      procedure
   }

clear();
_context.AbortOnError = 0;      set context so script does not exit on error,
                                allowing manual handling of errors.

CHOOSE INPUT IMAGE to be resampled

class RVC_OBJITEM riObjItem;                    objItem for input raster
DlgGetObject("Select raster to resample:", "Raster", riObjItem, "ExistingOnly");

PIPELINE SOURCE for input image

class IMAGE_PIPELINE_SOURCE_RVC source_In( riObjItem );
err = source_In.Initialize();
if (err < 0)
   ReportError(_context.CurrentLineNum, err);
else   print("Pipeline source initialized.");

printf("Source image has %d lines and %d columns.\n", source_In.GetTotalRows(),
        source_In.GetTotalColumns() );

check that input image source has valid coordinate reference system

class IMAGE_PIPELINE_GEOREFERENCE sourceGeoref;
sourceGeoref = source_In.GetGeoreference();

get coordinate reference system from the source georeference

class SR_COORDREFSYS crs;
crs = sourceGeoref.GetCRS();

if (crs.IsDefined() == 0 or crs.IsLocal() ) {
   PopupMessage("Source coordinate reference system is undefined or local;
         exiting script.");
   Exit();
   }
else   printf("Coordinate reference system: %s\n", crs.Name );

CHOOSE REFERENCE RASTER TO RESAMPLE TO

class RVC_OBJITEM refObjItem;
DlgGetObject("Select reference raster:", "Raster", refObjItem, "ExistingOnly");

SET PIPELINE SOURCE FOR REFERENCE IMAGE

class IMAGE_PIPELINE_SOURCE_RVC source_Ref( refObjItem );
err = source_Ref.Initialize();
if (err < 0)
   ReportError(_context.CurrentLineNum, err);
else
   print("Reference source initialized.");

printf("Reference image has %d lines and %d columns.\n",
   source_Ref.GetTotalRows(), source_Ref.GetTotalColumns() );

check that reference image has valid coordinate reference system

class IMAGE_PIPELINE_GEOREFERENCE refGeoref;
refGeoref = source_Ref.GetGeoreference();

get coordinate reference system from the source georeference

class SR_COORDREFSYS crsRef;                    reference raster's coordinate
crsRef = refGeoref.GetCRS();                    reference system

if (crsRef.IsDefined() == 0 or crsRef.IsLocal() ) {
   PopupMessage("Reference coordinate reference system is undefined or local;
         exiting script.");
   Exit();
   }
else
   printf("Reference coordinate reference system: %s\n", crsRef.Name );

class RVC_OBJITEM rastOutObjItem;               CHOOSE OUTPUT RASTER
DlgGetObject("Choose raster for resampled output", "Raster", rastOutObjItem,
         "NewOnly");

get line and column cell sizes from input and reference images

class POINT2D scaleIn, scaleRef;        column and line cell sizes of input
                                        and reference as x and y values of
                                        POINT2D;
class POINT2D locIn, locRef;            column and line locations for
                                        which to obtain cell size
locIn.x = source_In.GetTotalColumns() / 2;      location at center
locIn.y = source_In.GetTotalRows() / 2;         of input image

locRef.x = source_Ref.GetTotalColumns() / 2;    location at center
locRef.y = source_Ref.GetTotalRows() / 2;       of reference image

sourceGeoref.ComputeScale(locIn, scaleIn, 1);   pixel scales of input and
refGeoref.ComputeScale(locRef, scaleRef, 1);    reference image in meters

pixel scale can be negative, so get absolute values

scaleIn.x = abs(scaleIn.x);   scaleIn.y = abs(scaleIn.y);
scaleRef.x = abs(scaleRef.x);      scaleRef.y = abs(scaleRef.y);
printf("Input image cell sizes: line = %.2f m, col = %.2f m\n", scaleIn.y, scaleIn.x);
printf("Reference image cell sizes: line = %.2f m, col = %.2f m\n", scaleRef.y,
   scaleRef.x);

SET APPROPRIATE RESAMPLING METHOD based on relative
cell size of input and reference (output) images

numeric inCellArea, outCellArea;
inCellArea = scaleIn.x * scaleIn.y;     outCellArea = colCellSize * lineCellSize;

string rsmpMethod$;                     set resampling method
if (outCellArea > 2 * inCellArea) then rsmpMethod$ = "Average";
   else rsmpMethod$ = "CubicConvolution";
printf("Resampling method: %s\n", rsmpMethod$);

PIPELINE FILTER to resample source image

class IMAGE_PIPELINE_FILTER_RESAMPLE filter_rsmp(source_In,
      source_Ref, rsmpMethod$);
err = filter_rsmp.Initialize();
if (err < 0)   ReportError(_context.CurrentLineNum, err);
else   print("Resample filter initialized.");

PIPELINE TARGET: set up the target for the pipeline

class IMAGE_PIPELINE_TARGET_RVC target_rvc(filter_rsmp, rastOutObjItem);
target_rvc.SetCompression("DPCM", 0);
err = target_rvc.Initialize();
if (err < 0)   ReportError(_context.CurrentLineNum, err);
else   print("Pipeline target initialized.");

print("Processing...");
target_rvc.Process();                   EXECUTE pipeline process

print("Done.");
```