

# Pipeline Image Processing

A *pipeline* is an efficient, modular software architecture commonly employed for tasks that can be broken down into a series of independent processing steps. MicroImages has integrated a pipeline image-processing architecture into the TNT products and the Geospatial Scripting Language (SML), where it can be used in combination with the wide array of other SML functions and classes.

### Pipeline Stages

A pipeline consists of a chain of processing elements arranged so that the output of each element (*stage*) is the input of the next. There are three types of stages (see more complete definitions in the box to the right): *source* (image input), *filter* (processing element), and *target* (image output). Sources and targets can be raster objects in a MicroImages Project File, or files in other formats supported for direct use in the TNT products (see lists of source and target types below). Filters are provided to perform a variety of operations such as resampling, mosaicking, applying spatial filters, cropping, applying a mask, and many others (see list of filters on the reverse).

### Pipeline Connections and Operation

Each type of source, filter, and target is a separate SML class with its own predefined properties and methods (class functions). Pipeline connections are forged when a stage class is constructed in a pipeline script by specifying the previous stage that provides its input as part of its definition. A pipeline can have one or several sources, but only one target. Filters can be applied in series to one image or in parallel to multiple source images. Once the pipeline is constructed, a single method is called on the target stage to initiate processing and pull all of the image data through the pipeline. Some examples of simple pipeline designs are diagrammed below.

### Pipeline Benefits

Pipeline stages encapsulate their data, data properties, and operations. They also interact with each other in simple, defined ways. This modular design simplifies coding in SML and makes it easy to construct, modify, or extend a processing pipeline in a script. For example, in an SML pipeline georeference information is an inherent property of an image, so it is automatically pulled through the pipeline and assigned to the target. Likewise, pyramid tiers are automatically produced for target rasters in Project Files. Scripts run on single and multi-core computers automatically use any multi-threading incorporated into the stages such as in JPEG2000 compression and decompression operations.

Pipeline Source Types:	
RVC	Raster object in MicroImages Project File
PNG	PNG file
TIFF	TIFF file
JPEG	JPEG file
MRSID	MrSID file
WBMP	Wireless Bitmap (WBMP) file
REGION	Region to use for masking/cropping
CONSTANT	Source with constant value

more sources  
by request

Pipeline Target Types:	
RVC	Raster object in MicroImages Project File
RVC_MULTIFILE	MicroImages tileset
J2K	J2K (JPEG2000) file
TIFF	TIFF file
PNG	PNG file
JPEG	JPEG file

more targets  
by request

### Pipeline Terminology

**IMAGE:** a raster object, file, or equivalent structure in memory consisting of one component / band, or a set of co-registered components / bands. If there is more than one component, each has the same DIMENSIONS (total number of rows and columns), data type, and georeference. Examples: an elevation raster object in a Project File, an RGB color-composite raster object in a Project File, or a GeoTIFF file containing four bands of an Ikonos or QuickBird satellite scene.

**SAMPLE:** the numeric value for a particular image row/column position and component. A sample has a Data Type property (e.g. unsigned 8-bit, signed 16-bit, 32-bit floating-point, and so on).

**PIXEL:** the set of SAMPLES (one sample per component) for a particular image row/column position. A PIXEL has a Pixel Type property that specifies the number and relationship (if any) of its SAMPLES (e.g. grayscale, multiple, RGB, CMYK, and so on).

**STAGE:** any pipeline element that represents or processes an image.

**SOURCE:** a pipeline stage that inputs an image. A source stage has no pipeline inputs and one output.

**FILTER:** a pipeline stage that applies some processing or transformation to the image. A filter stage has one or more inputs and one output.

**TARGET:** a pipeline stage that represents the final output image. A target has one input and no pipeline output. Its properties are derived from the input stage it is connected to.

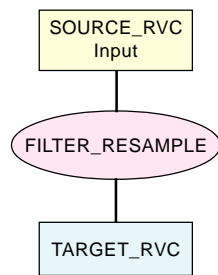
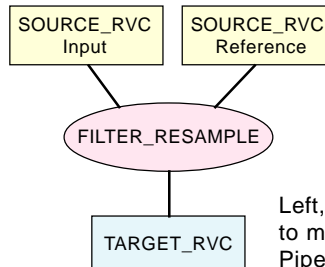
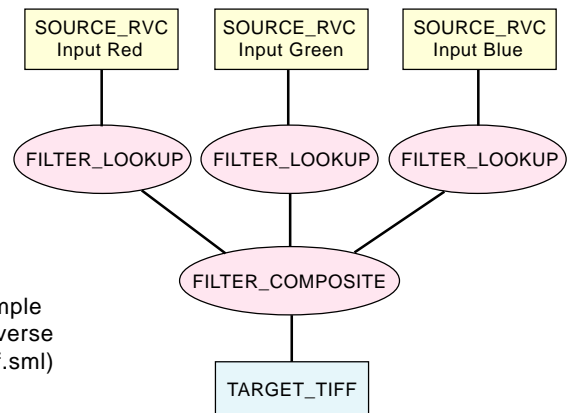


Diagram of pipeline to resample an image in the script on reverse (PipelineResampleToGeoref.sml)



Left, diagram of pipeline to resample an image to match a reference image (see sample script PipelineResampleToMatch.sml).



Above, diagram of pipeline to apply contrast to red, green, and blue grayscale images, make a color composite, and output it to a TIFF file (see sample script PipelineContrastCompositeToTiff.sml).

## Pipeline Filters in MicroImages Geospatial Scripting Language (SML)

COLORBALANCE color-balance an image  
 COMPOSITE create composite from multiple sources  
 CROP crop an image  
 DATATYPE change sample data type  
 DEINDEX change indexed-color to separate samples  
 DIVIDE divide one image by another  
 EXTEND extend image by adding null pixels around outside  
 FOCAL\_TOPOGRAPHIC compute topographic properties  
 FOCAL\_AMPM Adaptive Mean P-median spatial filter  
 FOCAL\_CONTRAST Locally-adaptive contrast spatial filter  
 FOCAL\_CS Comparison/selection spatial filter  
 FOCAL\_FROST Frost radar noise reduction spatial filter  
 FOCAL\_GRADIENT gradient edge-detection filter  
 FOCAL\_KUANADAPTIVE Kuan radar noise reduction filter  
 FOCAL\_LEE Lee radar noise reduction filter  
 FOCAL\_LUM lower-upper-middle spatial filter  
 FOCAL\_MLM multilevel median spatial filter  
 FOCAL\_MULTIPLICATIVE weighted averaging filter  
 FOCAL\_OLYMPIC olympic spatial filter  
 FOCAL\_PMEDIAN P-median spatial filter  
 FOCAL\_RANGE range spatial filter  
 FOCAL\_SIGMA sigma radar noise reduction filter  
 FOCAL\_STATISTICS compute focal mean, median, sum, etc.  
 FOCAL\_TEAGER Teager spatial filter  
 FOCAL\_VOLTERRA Voltterra-unsharp spatial filter  
 FOCAL\_WMMR weighted-majority/minimum range-median filter  
 FUSION\_BASICHBS multiresolution image fusion via HBS  
 FUSION\_BASICHIS multiresolution image fusion via HIS  
 FUSION\_BROVEY multiresolution image fusion via Brovey method  
 FUSION\_MODHIS multiresolution image fusion via modified HIS  
 FUSION\_TEXTUREPCA multiresolution image fusion via TexturePCA

more filters  
by request

LINEAR perform linear combination on samples in image  
 LOOKUP apply look-up tables to samples in image  
 MASK\_SELECT select between images using mask  
 MASK\_VALIDITY apply mask to image  
 MORPHOLOGY\_CLOSING morphological operation "closing"  
 MORPHOLOGY\_DILATION morphological operation "dilation"  
 MORPHOLOGY\_EROSION morphological operation "erosion"  
 MORPHOLOGY\_OPENING morphological operation "opening"  
 MOSAIC mosaic images  
 MULTIPLY multiply images  
 OVERRIDETYPE override pixel type of image  
 PIXELTABLE apply pixeltable (e.g. color palette) to image  
 PIXEL\_TYPE change pixel type of image  
 QUANTIZE quantize samples in image  
 REPLACE\_NULL replace null pixels with specified value  
 RESAMPLE resample/reproject image  
 SCALEOFFSET apply scale and offset to samples  
 SELECT select specified component samples from image  
 STEREO generate stereo image  
 TESTEQUAL test pixel for equality  
 TESTRANGE test pixel for containment inside or outside range  
 VALIDITYNEAR set validity based on nearness to specified value  
 ZOOM zoom image (change pixel size) up or down by fixed amount

### Special Filters that Apply Pixel-by-Pixel Computations using a User-Defined Delegate Function

GENERAL\_CHGPROP apply user-defined function to image with change of pixel type  
 GENERAL\_INPLACE apply user-defined function to image  
 MARGIN automatically buffer margins when iterating with user-defined function

## Pipeline Script to Resample/Reproject Image to Specified Cell Size: PipelineResampleToGeoref.sml

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

error checking procedure

CHOOSE INPUT RASTER to be resampled

```
class RVC_OBITEM riObjItem;
DlgGetObject("Select raster to resample:", "Raster", riObjItem, "ExistingOnly");
```

objItem for input raster

PIPELINE SOURCE: set input raster as source

```
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 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 OUTPUT RASTER

```
class RVC_OBITEM rastOutObjItem;
DlgGetObject("Choose raster for resampled output", "Raster", rastOutObjItem,
    "NewOnly");
```

```
get line and column cell sizes from source's georeference
```

```
class POINT2D scaleIn;
class POINT2D locIn;
```

line and column cell sizes as x and y values of POINT2D; column and line location for which to obtain cell size

```
locIn.x = source_In.GetTotalColumns() / 2;
locIn.y = source_In.GetTotalRows() / 2;
sourceGeoref.ComputeScale(locIn, scaleIn, 1);
printf("Source image cell sizes: line = %.2f m\n", scaleIn.y, scaleIn.x);
```

location at center of image

pixel scales in meters

prompt user to enter desired output line/column cell sizes

```
numeric lineCellSize, colCellSize;
```

cell size to set for the output raster

```
string prompt$ = "Enter desired line cell size for output raster:";
lineCellSize = PopupNum(prompt$, scaleIn.y, 0, 1000, 2);
prompt$ = "Enter desired column cell size for output raster:";
colCellSize = PopupNum(prompt$, scaleIn.x, 0, 1000, 2);

[code to compute appropriate resampling method omitted]
```

PIPELINE FILTER to resample source image

```
class IMAGE_PIPELINE_FILTER_RESAMPLE filter_rsmp(source_In, crs,
    lineCellSize, colCellSize, 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();
print("Done.");
```

EXECUTE pipeline process