

Автоматизация выполнения моделей в ERDAS IMAGINE

[Обсудить в форуме](#) Комментариев — 5

Эта страница опубликована в основном списке статей сайта по адресу <http://gis-lab.info/qa/batch-erdas.html>

Экономим время при сохранении четкой логики процесса обработки групп изображений.

Допустим есть модель, которую мы хотим автоматизировать. Допустим что эта модель использует 2 файла на входе, назовем их А и В и один — на выходе С. Схематично операция будет выглядеть так:

$A + B + \dots = C$

Исходные растры А и В могут быть исходными данными для сколь угодно сложных вычислений, которые можно реализовать в инструменте моделирования ERDAS. Более того, исходных растров может быть больше чем 2. Проиллюстрируем процесс на простом сложении двух растров, для простоты объяснения использовать будем только их первые каналы.

Наша задача состоит в том, чтобы используя эту модель преобразовать наборы файлов А и набор файлов В в результат — набор файлов С.

Продemonстрируем по шагам выполнение данной автоматизации.

Содержание

- [1 Экспортируем модель \(файл с расширением gmd\) в скрипт \(файл с расширением mdl\)](#)
- [2 Заменяем меняющиеся параметры в модели \(имена файлов\) на аргументы](#)
- [3 Используем пакетный процессор для запуска модели с аргументами](#)
- [4 Видоизменим bsc-файл введя параметры и туда](#)
- [5 Выберем обрабатываемые изображения](#)
- [6 Ссылки по теме](#)

Экспортируем модель (файл с расширением gmd) в скрипт (файл с расширением mdl)

Для этого, открываем модель в Model Maker и выбираем «Process → Generate script». Содержание подобного скрипта может выглядеть например вот так:

```
COMMENT "Generated from graphical model: c:/temp/test.gmd";
SET CELLSIZE MIN;
SET WINDOW UNION;
SET AOI NONE;
#
# declarations
#
Integer RASTER n1 FILE OLD NEAREST NEIGHBOR AOI NONE "c:/temp/imageA.img";
Integer RASTER n2 FILE OLD NEAREST NEIGHBOR AOI NONE "c:/temp/imageB.img";
Integer RASTER result FILE DELETE_IF_EXISTING IGNORE 0 ATHEMATIC 16 BIT SIGNED INTEGER
"c:/temp/imageC.img";
#
# function definitions
#
#
define func Integer($n1(1) + $n2(1))
result = func;
```

```
QUIT;
```

Изучим содержимое скрипта, не обращая внимания на то, что именно он делает, но попытаемся разобраться как происходит адресация файлов с которыми он работает. Как можно видеть из примера — файлы указываются в явном виде в виде полного к ним пути. Выполнить такой скрипт легко можно с помощью Model Librarian, однако он выполнит операцию только для той пары файлов, которая указана в содержимом скрипта. К сожалению, возможности создать массив из имен новых файлов в таком скрипте нет. Да если бы и была, при новом наборе файлов для обработки пришлось бы переписывать их имена заново в этот массив.

Заменим меняющиеся параметры в модели (имена файлов) на аргументы

Аргументы — специальные кодовые слова, которые сможет использовать пакетный процессор ERDAS. Наш файл приобретет следующий вид, имена файлов замененные на аргументы выделены жирным:

```
COMMENT "Generated from graphical model: c:/temp/test.gmd";
SET CELLSIZE MIN;
SET WINDOW UNION;
SET AOI NONE;
#
# declarations
#
Integer RASTER n1 FILE OLD NEAREST NEIGHBOR AOI NONE ARG1;
Integer RASTER n2 FILE OLD NEAREST NEIGHBOR AOI NONE ARG2;
Integer RASTER result FILE DELETE_IF_EXISTING IGNORE 0 ATHEMATIC 16 BIT SIGNED INTEGER
ARG3;
#
# function definitions
#
#
define func Integer($n1(1) + $n2(1))
result = func;
QUIT;
```

Уже лучше, теперь нужно найти способ подставить в эти аргументы наши имена файлов.

Используем пакетный процессор для запуска модели с аргументами

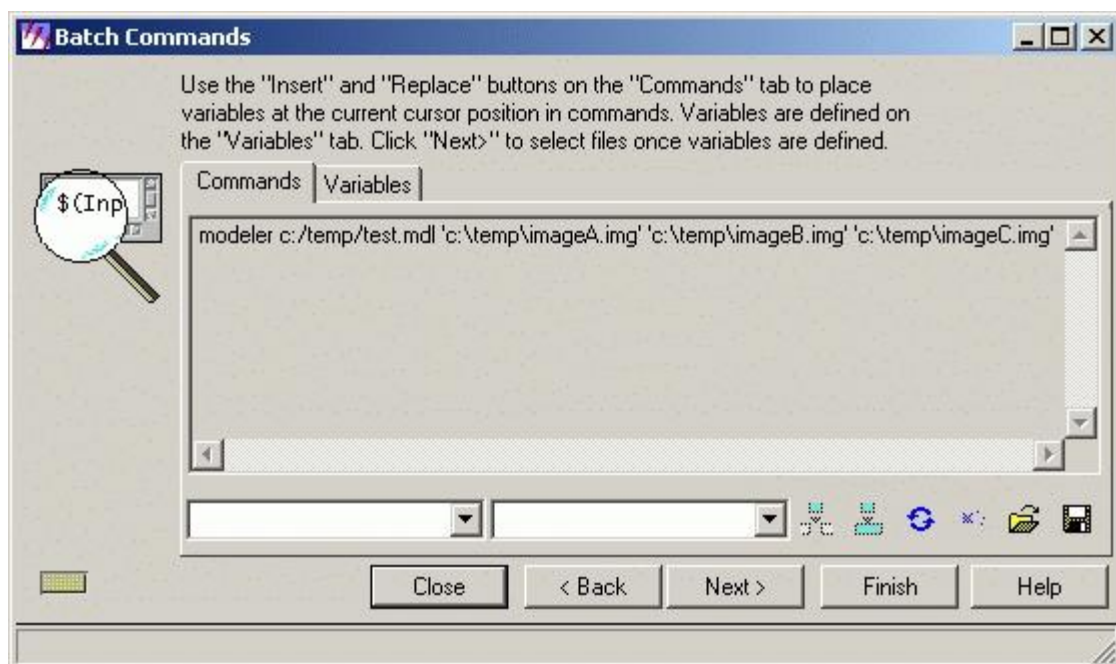
Перед этим создадим batch-файл (test.bcf), который будет запускать модель, такого содержания:

```
modeler c:/temp/test.mdl c:\temp\imageA.img c:\temp\imageB.img c:\temp\imageC.img
```

Как видно из содержания, процесс запуска модели состоит из передачи таких параметров как имя модели и трех аргументов специальной программе — modeler'y.

Уже лучше, так как нам не нужно хранить файлы в самой модели, однако все еще не очень хорошо, так как теперь вместо того, чтобы генерировать 100 моделей для 100 пар исходных файлов, надо генерировать 100 записей в bcf-файл с разными аргументами.

Обращаем внимание, что возможности подстановки параметров в bcf в пакетном процессоре присутствует, это видно например из этого примера, однако в нашем случае ничего в полях параметров выбрать нельзя, потому что параметры нужно определить еще и в bcf-файле.



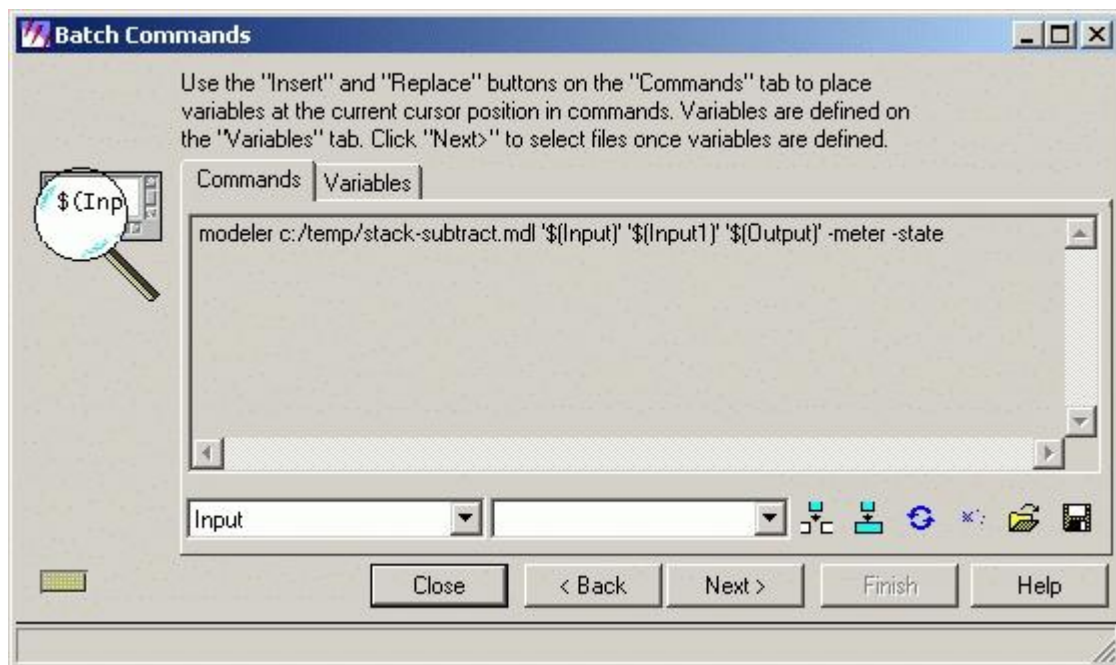
Идем дальше.

Видоизменим bcf-файл введя параметры и туда

Новый bcf будет выглядеть таким образом:

```
variable Input User;
variable Input1 User;
variable Output Auto "c:/temp//$(Input.root).img" Delete_Before;
modeler c:/temp/test.mdl '$(Input)' '$(Input1)' '$(Output)' -meter -state
```


Загрузим bcf-файл в пакетный процессор, отметим, что появилась возможность выбора параметров (ими в нашем случае будут имена файлов).

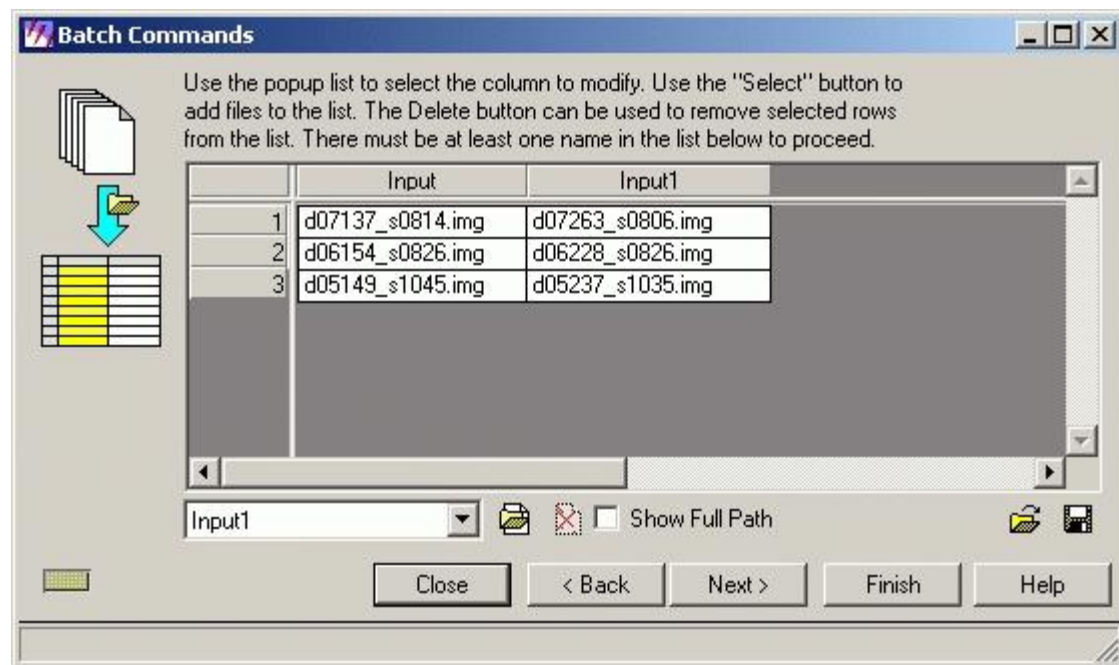


Выберем обрабатываемые изображения

Последним шагом нашей автоматизации будет выбор собственно изображений, для этого нажмем «Next» и появившемся окне сначала выберем группу изображений в качестве ImageA и группу в качестве ImageB, не забывая менять название параметра которому мы подставляем исходные растры.

Удобным способом выбора будет сначала выбрать в параметрах Input и выбрать с помощью кнопки

 («Select files to add») серию первых исходных растров (imageA), а затем выбрать в параметрах Input1, выделить колонку Input1 и выбрать с помощью той же кнопки серию вторых исходных растров (imageB).



Автоматизация завершена, осталось только выполнить пакет — «Finish».

Ссылки по теме

- [Назначение системы координат в пакетном режиме в ERDAS IMAGINE](#)

[Обсудить в форуме](#) Комментариев — 5

Последнее обновление: 2014-05-15 01:17

Дата создания: 08.12.2007

Автор(ы): [Максим Дубинин](#)