Creating Faster, more Expressive ImageJ Scripts and Plugins with Scala

Jarek Sacha

IJ-Plugins Project / Procter & Gamble

USA

jsacha@users.sf.net http://ij-plugins.sf.net/

Abstract

ImageJ libraries and plugins are traditionally developed using Java. Java provides execution speed and a programming environment suitable for creation of large code bases.

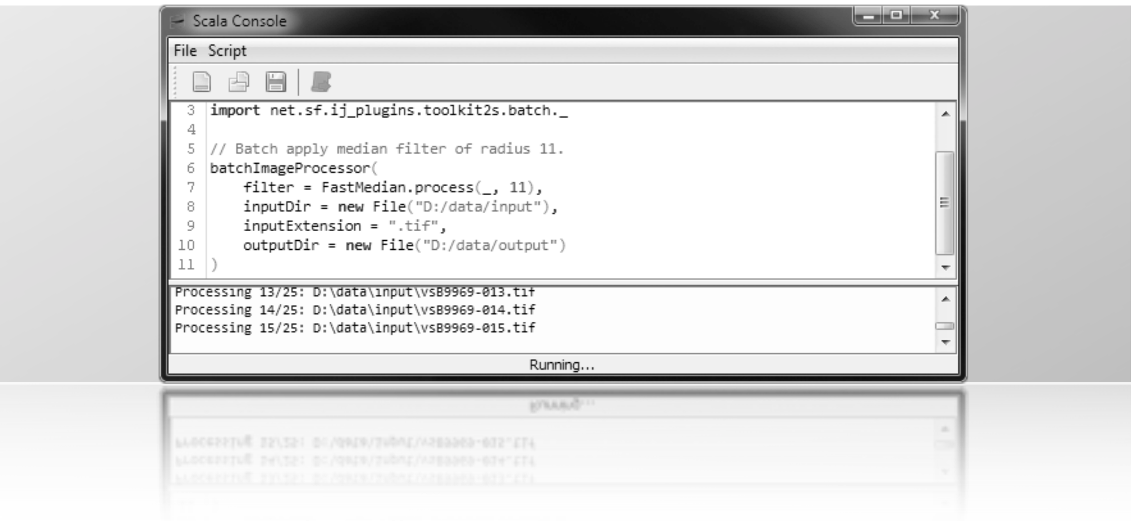
Smaller, more expressive, code is created using ImageJ macros and scripting in dynamic languages like JavaScript or Groovy. Macros/scripts help quickly create specialized code with minimal boiler-plate, in syntax more expressive than Java. Main drawbacks are a much slower execution speed and a need to learn/use two different types of syntax for scripting and plugin/library code.

Scala is a statically-typed general purpose language that has a feel of a dynamic language. You can create code that is as expressive as a dynamic language but executes at the speed of compiled Java code. It can be used both for scripting, without loss of execution speed, and for “regular” library code. Scala code integrates with existing Java libraries like ImageJ. Java can also call Scala libraries. Scala builds on strength of Java, adding many advanced language features like closures, traits, type inference, powerful generics and concurrency support, and many more.

This presentation will show how Scala can be used to create image analysis libraries and scripts that are more expressive (smaller number of lines and characters) then existing Java code, at the same time being at least as fast or faster. Scala annotations can be used to actually generate code that is more computationally efficient than equivalent Java code. Specialized compiler plugins can be used to further optimize the byte code and to take advantage of dedicated hardware, like GPUs.

Keywords

Image analysis, image processing, ImageJ, Scala, scientific computing, software development



```
import net.sf.ij_plugins.toolkit2s.batch._

// Batch apply median filter of radius 11.
batchImageProcessor(
    filter = FastMedian.process(_, 11),
    inputDir = new File("D:/data/input"),
    inputExtension = ".tif",
    outputDir = new File("D:/data/output")
)
```

```
Processing 13/25: D:\data\input\vsB9969-013.tif
Processing 14/25: D:\data\input\vsB9969-014.tif
Processing 15/25: D:\data\input\vsB9969-015.tif
```