

# ADAMS

Advanced **D**ata mining **A**nd **M**achine learning **S**ystem

Module: adams-imaging-imagej



Peter Reutemann

December 20, 2017

©2009-2015



THE UNIVERSITY OF  
**WAIKATO**  
*Te Whare Wānanga o Waikato*



Except where otherwise noted, this work is licensed under  
<http://creativecommons.org/licenses/by-sa/4.0/>

# Contents

<b>1</b>	<b>ImageJ</b>	<b>7</b>
1.1	Flow . . . . .	7
1.2	Plugins . . . . .	9
<b>2</b>	<b>Feature output</b>	<b>11</b>
<b>3</b>	<b>Object conversion</b>	<b>13</b>
	<b>Bibliography</b>	<b>15</b>



# List of Figures

1.1	ImageJ flow for turning images stored in a directory into greyscale ones. . . . .	8
1.2	The original image. . . . .	8
1.3	The greyscale image. . . . .	8
2.1	Generating a CSV file using ImageJ. . . . .	11
2.2	The ImageJ generated CSV file. . . . .	12



# Chapter 1

## ImageJ

ImageJ is a public domain software suite written in Java (using AWT, opposed to Swing which ADAMS uses) for image processing, developed at National Institutes of Health ([2]).

### 1.1 Flow

The following ImageJ actors available:

- `transformer.ImageJReader` – for reading any image file that JAI supports<sup>1</sup> and forwarding an `ImagePlusContainer` object.
- `transformer.ImageJTransformer` – performs a transformation using an existing ImageJ transformer class on the incoming image and outputs another image again. ImageJ plugin filters, commands and pre-recorded macros can be used to perform transformations.
- `transformer.ImageJFeatureGenerator` – turns an `ImagePlusContainer` into an `weka.core.Instance` object to be used in WEKA. The attached meta-data in form of a report can be added to the output object as well.
- `transformer.ImageJReleaseAllImages` – removes all images currently listed in ImageJ's batch mode list, freeing up memory.
- `sink.ImageJReleaseImage` – removes the incoming image from ImageJ's batch mode list of images, freeing up memory.
- `sink.ImageJWriter` – for writing an `ImagePlusContainer` to a file format that ImageJ supports. If the image type cannot be determined based on the extension, you can also specify which type to generate.

Figure 1.1 shows a flow<sup>2</sup> for reading images, turning them into greyscale using a transformer and displaying them side-by-side. Figures 1.2 and 1.3 show original and greyscale image.

---

<sup>1</sup>[http://imagejdocu.tudor.lu/doku.php?id=faq:general:which\\_file\\_formats\\_are\\_supported\\_by\\_imagej](http://imagejdocu.tudor.lu/doku.php?id=faq:general:which_file_formats_are_supported_by_imagej)

<sup>2</sup>`adams-imaging-transform.to-greyscale.flow`

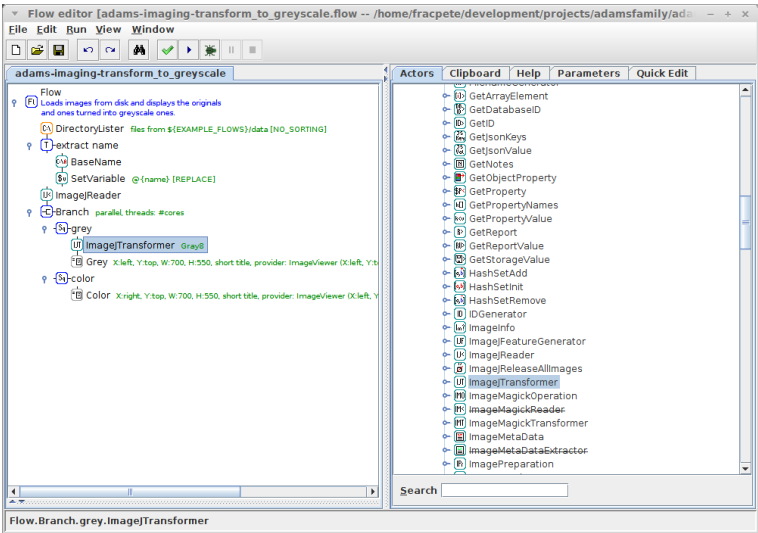


Figure 1.1: ImageJ flow for turning images stored in a directory into greyscale ones.

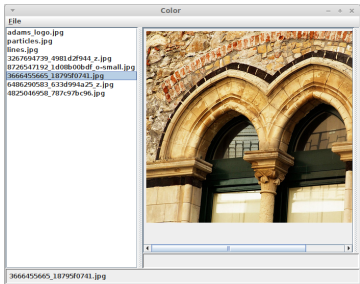


Figure 1.2: The original image.

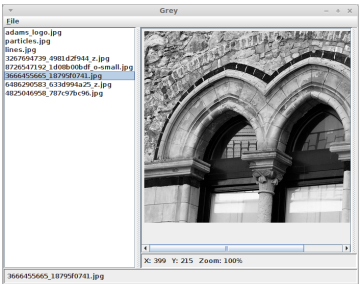


Figure 1.3: The greyscale image.



## 1.2 Plugins

By default, ADAMS includes plugins located in the following directory on Linux/Unix/Mac:

```
$HOME/.adams/imagej/plugins
```

and on Windows here:

```
%USERPROFILE%/_adams/imagej/plugins
```

You can override this directory by using the `ADAMS_IMAGEJ_DIR` environment variable, which defines the directory one level above the *plugins* directory. For instance, if your plugins directory is located at:

```
/home/user/imagej/plugins
```

You have to define the `ADAMS_IMAGEJ_DIR` environment variable as follows:

```
ADAMS_IMAGEJ_DIR=/home/user/imagej
```



## Chapter 2

# Feature output

Of course, the data can be turned into a format that is suitable for machine learning applications like WEKA ([?]). For JAI and ImageMagick transformers, both generating *BufferedImageContainer* tokens, the *BufferedImageFeatureGenerator* can be used to generate such output. For ImageJ generated tokens, outputting *ImagePlusContainer* tokens, you have to use the *ImageJFeatureGenerator* instead. What kind of output is generated, depends on the *feature converter* defined in those feature generator transformers. By default, spreadsheet data is generated, which can be stored in CSV files. Figure 2.1 shows a flow<sup>1</sup> that generates a CSV file from images using ImageJ. The resulting dataset, as displayed in the spreadsheet viewer, is shown in Figure 2.2.

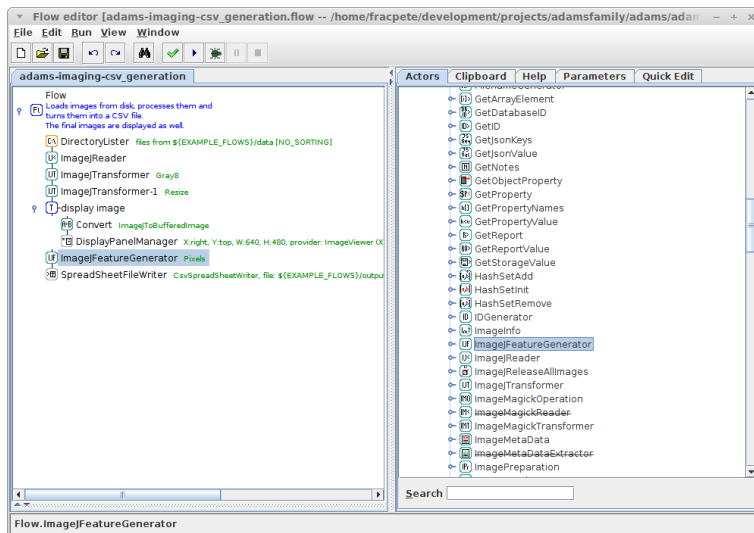


Figure 2.1: Generating a CSV file using ImageJ.

<sup>1</sup>adams-imaging-csv\_generation.flow

Row	att_1	att_2	att_3	att_4	att_5	att_6	att_7	att_8	att_9	att_10	att_11	att_12	att_13	att_14	att_15	att_16	a
1	A/1	B/2	C/3	D/4	E/5	F/6	G/7	H/8	I/9	J/10	K/11	L/12	M/13	N/14	O/15	P/16	a
2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	93	94	94	95	95	97	98	99	99	101	102	103	103	104	105	107	
6	66	71	75	79	33	17	78	82	79	60	80	81	77	73	70	10	
7	-96	-93	-71	88	-121	-96	-91	-95	-117	-124	-61	120	-84	110	-64	-117	
8	67	20	10	17	13	26	33	47	58	53	-72	-53	-69	-51	-49	-70	
9	3	4	3	4	3	3	3	4	3	3	4	4	6	6	7	6	

Figure 2.2: The ImageJ generated CSV file.

## Chapter 3

# Object conversion

The following conversions are available to convert from one format into another:

- *ImageJToBufferedImage* – converting from ImageJ to JAI/ImageMagick.



# Bibliography

- [1] *ADAMS* – Advanced Data mining and Machine learning System  
<https://adams.cms.waikato.ac.nz/>
- [2] *ImageJ* – Image Processing and Analysis in Java  
<http://rsbweb.nih.gov/ij/>