Diva-DID : Document Images Degradation

Mathias Seuret

October 20, 2015

Contents

1 Introduction

DIVA-DID has hard-coded image manipulation methods which can be combined together in easy-to-write scripts. The scripts are stored in XML files and offer simple methods for loading, writing and manipulating document images. This documents presents *very* briefly how to use it.

The application can be started with the following command:

Listing 1: Starting the application

```
1  java -jar DivaDid.jar script-file.xml [parameters]
```

The parameters are variable definitions, for example to indicate which file has to be processed by a script.

In the following parts of this user guide, we will present how to run scripts, how to print text in the terminal, how to use variables, how to load/save images and finally the different image manipulation methods.

2 Basic XML script

The script files are stored in an XML format. The root element can have any name but should contain an ID, which will be displayed along error messages – this makes error tracking easier. The tag contained by the root element correspond to commands, they are executed sequentially. The smallest possible script file is

Listing 2: Smallest script file

```
1 <do-something id="test">
2   <!-- add content here -->
3 </do-something>
```

3 Printing text

The tag "print" can be used for outputting stuff, e.g., the current step being done by the script.

Listing 3: Printing text

```
1 <do-something id="test">
2   <print>This is the first line</print>
3   <print>This is the second line</print>
4 </do-something>
```

The text is first pre-processed and the aliases (see Section 4) are replaced by their value.

4 Variables - aliases

Variables do not really exist in the script, but there is something similar to it inspired by the "#define" of the C pre-processor: the aliases. An alias is a string

(key) which is automatically replaced by another string (value) in the script, with the exception of tag names and definitions of new aliases.

Aliases can be defined in two ways. The first way is to use the "alias" tag:

Listing 4: Defining an alias in the script

```
<my-script id="test">
  <alias id="MY-ALIAS" value="123" />
  <print>My alias has the value MY-ALIAS</print>
</my-script>
```

The second way is to define them from the command line:

Listing 5: Definint an alias from the CLI

```
  java -jar did.jar script-file.xml MY-ALIAS="Hello World !"
```

If the attribute "value" is not defined in the alias tag, then it will check if the alias has been defined from the CLI. If it has not, then an exception will be thrown.

5 Loading, creating & saving images

Images can be loaded/created with the "image" tag:

Listing 6: IO with images

```
<test id="test">
  <alias id="INPUT" value="my-image.jpg" />
  <image id="my-image">
    <load file="INPUT" />
  </image>
  <image id="my-copy">
    <copy ref="my-image" />
  </image>
  <!-- do stuff -->
  <delete ref="my-image" />
  <!-- do some more stuff -->
  <save ref="an-existing-image" file="output.jpg" />
</test>
```

An ID is attributed to each image and they are stored internally in a map. When an image will not be used anymore in a script, it has to be deleted manually if you want to free some memory. The images are stored as 3D arrays of floats, therefore they use quite a lot of memory.

Commands using images will always refer to them using the ID which is defined when they are created.

6 Degrading images

Several methods for adding degradations to document images are presented in the following subsections. They give better results when they are combined together.

6.1 Pepper and salt noise

The pepper and salt noise changes the value of random pixels to either black or white. The tag "pepper-and-salt" requires a reference to the image to noise, and it must contain a "fraction" child indicating which fraction of the pixels should be modified.

Listing 7: Pepper and salt noise

```
<pepper-salt ref="image-id">
  <fraction>0.35</fraction>
</pepper-salt>
```

6.2 Gaussian noise

The Gaussian noise adds to all pixels random values following a Gaussian distribution. The random values r_i are computed with

$$r_i = \alpha \cdot \sqrt{2 \cdot \log\left(\frac{1}{R_i}\right)}$$

where R_i is a random value in $[0; 1[$. The factor α corresponds to the strength of the noise - a high value will noise more than a low value.

Listing 8: Gaussian noise

```
<gaussian-noise ref="image-id">
  <strength>0.1</strength>
</gaussian-noise>
```

6.3 Gradient-domain degradations

Gradient-domain degradations are based on the modification of the 6 gradients of a document image (horizontal and vertical gradients for all three RGB channels) followed by the reconstruction of the image from the modified gradients. The reconstruction requires many iterations and this is time consuming but gives results much more realistic than salt and pepper or Gaussian noise.

This degradation method pastes in the gradient domain patches of noise coming from other documents. This adds the patches in a rather realistic way, and does not require the noise patches to have the same background color or texture as the target document. Some artifacts in the dark parts of the document can be removed with the command presented in section 7.3.

There are two main ways to apply these degradations. Either, the user select a folder containing degradations and indicates a strength and a density, or the user indicates manually[1] where each degradation should be placed, and with which strength.

The syntax for the first case is:

Listing 9: Gradient-domain noise

```
<gradient-degradations ref="noised">
  <strength>1.2</strength>
```

[1] Or with a program using a noise model.

```
  <density>25</density>
  <iterations>750</iterations>
  <source>data/spots</source>
  <gpu/>
</gradient-degradations>
```

where:

- Strength: multiplication factor of the intensity of the patches,
- Density: average number of patches covering any pixel of the document
- Iterations: for the reconstruction
- Source: folder containing the patches
- Gpu: indicates that the reconstruction should be done with the GPU
  - The GPU can be selected: <gpu>num</gpu>
  - The processor can also be used, by replacing <gpu> by either
    - <single-core>
    - <multi-core>

For the second case, the syntax is
The syntax for the first case is:

Listing 10: Manual gradient-domain degradation

```
<manual-gradient-degradations ref="img">
  <degradation>
    <file>stains/stain1.png</file>
    <strength>1.1</strength>
    <x>100</x>
    <y>200</y>
  </degradation>
  <degradation>
    <file>stains/stain2.png</file>
    <strength>0.8</strength>
    <x>200</x>
    <y>300</y>
  </degradation>
  ...
  <single-core/>
  <iterations>NB</iterations>
</manual-gradient-degradations>
```

7 Other commands

In addition to the degradation methods, other tools are available.

7.1 Selective blur

This bluring method computes for every pixel the average value of its neighbors (of a given range) which have values closer to the target pixel than a given threshold.

Listing 11: Selective blur

```
<selective-blur ref="my-image">
  <range>4</range>
  <threshold>0.25</threshold>
</selective-blur>
```

7.2 Comparing two images

Several methods for comparing two images are provided.

Mean difference: the average difference between all components of all pixels of the images.

Listing 12: Mean absolute difference

```
<bas-diff>
  <a>my-first-image</a>
  <b>my-second-image</b>
</abs-diff>
<result red="COMP_RESULT" />
```

Mean square difference: similar to the mean difference, but with the square of the differences.

Listing 13: Mean absolute difference

```
<square-diff>
  <a>my-first-image</a>
  <b>my-second-image</b>
</square-diff>
<result red="COMP_RESULT" />
```

Scale and offset invariant mean square difference: similar to the mean difference, but with the square of the differences.

Listing 14: Mean absolute difference

```
<square-diff>
  <a>my-first-image</a>
  <b>my-second-image</b>
</square-diff>
<result red="COMP_RESULT" />
```

Gradient difference: compares the gradients of the images for the three channels. For each channel of each pixel, the gradient is a 2D vector ; this method compares the weighted mean of the angles of these vectors, using their length as weight.

Listing 15: Weighted gradient difference

```
<grad-diff>
  <a>my-first-image</a>
  <b>my-second-image</b>
</grad-diff>
<result red="COMP_RESULT" />
```

Correlation: The correlation between two images can be an useful measurement for generating synthetic data for machine learning, as we want the samples to have a low correlation.

Listing 16: Correlation of two images

```
<correlation>
  <a>my-first-image</a>
  <b>my-second-image</b>
</correlation>
<result red="COMP_RESULT" />
```

7.3 Luminosity-based mix

It seems that dark parts of the document suffer less from degradations due to the ink saturation. If there is some dark dirt on an ink layer, it will probably be less visible than on the background. Moreover, degradations in the gradient domain may make some areas reach negative luminosity when adding dark spots on inked areas. This command combines two images, ideally the original image and its degraded version, so that the darkest areas come from the original version.

Listing 17: Mixing two images in regard to their luminosity

```
<mix-images ref="target-image">
  <original>original-image</original>
</mix-images>
```