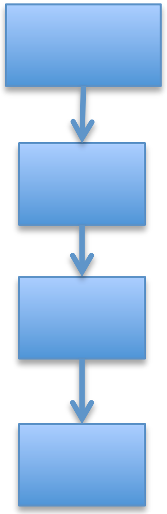
JT Batch Projects

PAR/NIST Cooperative

Eric Robertson

Adhoc Batch Tool

- Works in three modes:
  1. Start with initial images -> create projects
  2. Start with projects -> run plugin on last image found in each project
  3. Start with projects and images with same name as project -> connect image to last image of each project labeling with given operation information
- Dis-advantages?
  1. Produces single line of manipulations projects, thus it cannot support Paste Splice.
  2. Need to run all projects in collective stages, rather start and finish one project at a time.
- Advantages?
  1. Alleviates need to craft a project descriptor.
  2. Does not introduce randomness to image selection.
  3. Can extend existing projects

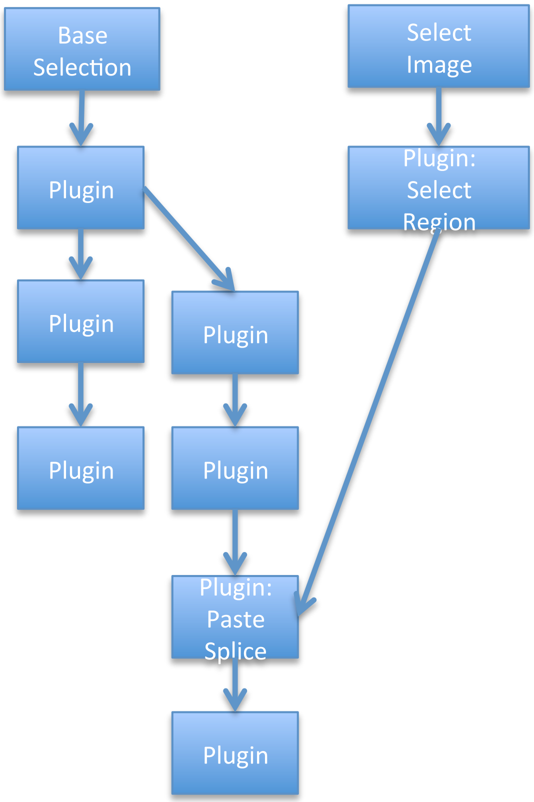


Batch Projects

- Unlike the Adhoc Batch Tool, Batch Projects creates a complete projects from start to finish.
- Batch Projects performs all manipulations with plugins.
- Batch Projects selects images from a pool of images (without replacement).
- The Batch Project graph mirror the final JT projects. What's the difference?
  - A Node is a plugin (operation)
  - A link is a dependency, defines order of operation.

Batch JT

Process Graph



Operation Nodes:

Select Image: Provided a pool (directory) of images

Base Selection: Same as Select Image, except start new project using the selected image name.

Plugin: Execute a plugin operation to create a new manipulated image. *Requires consistent parameters.*

Input Mask Plugin: Execute a plugin on a image to produce an (input) mask and any additional name/value pairs to be used to as parameters in subsequent operations. Input masks highlight groups of pixels (full intensity) as constraints to other plugins.

Operation Parameters :

- Includes donor images, fixed values, masks, etc.

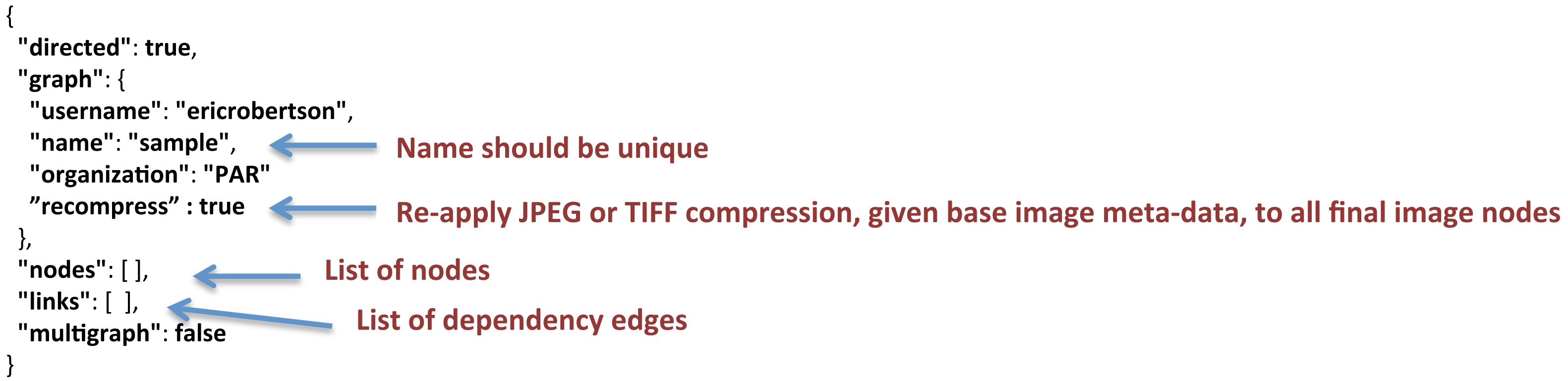
Links (edges):

- Form the dependency tree and order operation execution.

Each produced journal graph reflects the same structure as the process graph.

JSON

SHELL



```
{
 "directed": true,
 "graph": {
  "username": "ericrobertson",
  "name": "sample",
  "organization": "PAR"
  ”recompress” : true
 },
 "nodes": [ ],
 "links": [  ],
 "multigraph": false
}
```

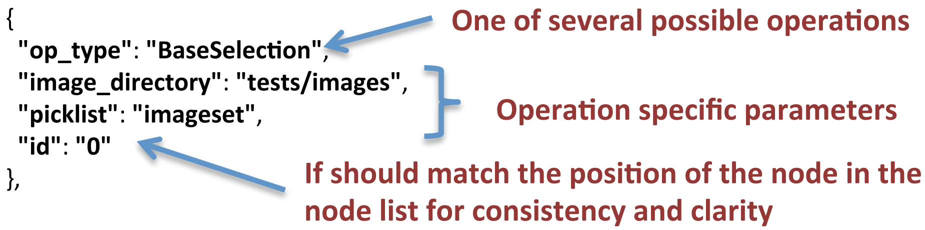
Name should be unique

Re-apply JPEG or TIFF compression, given base image meta-data, to all final image nodes

List of nodes

List of dependency edges

Node



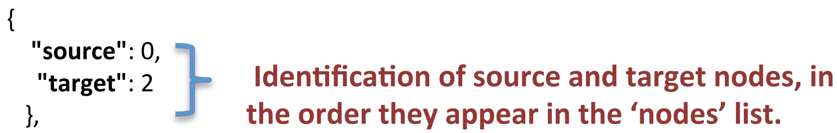
```
 {
  "op_type": "BaseSelection",
  "image_directory": "tests/images",
  "picklist": "imageset",
  "id": "0"
 },
```

One of several possible operations

Operation specific parameters

If should match the position of the node in the node list for consistency and clarity

Link



```
{
   "source": 0,
    "target": 2
  },
```

Identification of source and target nodes, in the order they appear in the ‘nodes’ list.

Operations

- **BaseSelection**

  Select an image from a pool of images maintained in a directory. The base selection provides a single image used to start a new project. The selected image provides the name of the project. The image is expected to serve as a (the) base image for the project that is to be manipulated. There must be one and only one BaseSelection node; it must not have any predecessor nodes.

  Parameters:

  - **"image_directory"** = a directory of a pool of images to select from (randomly)
  - **"picklist"**: an in memory structure tracking the names of image files already picked from projects to prevent future selection. A file is created with the same name in the 'working directory', retaining the pick list selection across multiple independent and sequential batch runs.

- **ImageSelection**

  Select an image from a pool of images maintained in a directory. Like BaseSelection, an ImageSelection node most not have any predecessor nodes. Unlike BaseSelection, multiple ImageSelection nodes are permitted.

  Parameters:

  The parameters are the same as the BaseSelection. The image directory and picklist can use the same pool as other ImageSelection and BaseSelection nodes.

Operations

- **PluginOperation**

  Invoke a plugin

  - Produce a manipulated target image given a source image.

  Parameters:

  - **"plugin" =** the name of the plugin
  - **"arguments" =** the set of arguments to be provided to the plugin. Each argument as a type and supporting descriptions. <u>Arguments fill **both** the requirements of the plugin and the requirements of the operation definition</u>.

- **InputMaskPluginOperation**

  Invoke a plugin

  - Produce an input mask and any additional name/value pairs based on the source image.

  Parameters:

  - **"plugin" =** the name of the plugin
  - **"arguments" =** the set of arguments to be provided to the plugin. Each argument as a type and supporting descriptions. <u>Arguments fill **both** the requirements of the plugin and the requirements of the operation definition</u>.

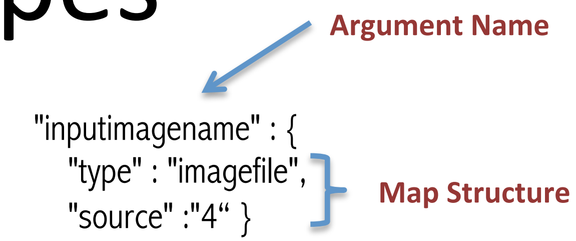
Plugin Review

- Plugins are operations.
- Plugins are provided:
  - Filename of source image
  - Filename of target image
  - Additional arguments
- Plugins action:
  - Overwrite target image
  - Optionally return name/value pairs that may be used as to set parameters of subsequent plugins.
- What is special about the input mask plugin?
  - Given a source image, pre-selects a group of pixels for alteration but subsequent operations.

Sample Plugins

Name	Description	Additional Parameters
SelectRegion	Select a region from a source image. Add an alpha channel, setting the unselected pixels to 0.	
PasteSplice	Place a selected region in a source image. Try to paste in area with the least amount of variance. Resize and rotate the selected region as necessary to fit into the selected area.	“Donor” image
SaveAsPNG	Save source image as a PNG. If the source image has EXIF metadata that contains Orientation and the parameter ‘Image Rotated’ is yes, rotate the image.	“Image Rotated”
GaussianBlur	Blur the entire image or a selected region of the image, given an optional input mask.	“kernelsize” is a tuple (x,y). The default value is (5,5). “inputmaskname” is the name of a monochrome image file where black pixels indicate the region to blur. The default is blur the entire image

Argument Types



```
"inputimagename" : {
   "type" : "imagefile",
   "source" :"4“ }
```

Argument Name

Map Structure

Each argument is a map containing a set of properties including a type.

- **imagefile = select an image produced by another node. The source node is provided using it’s node id.**

```
{
   "type" : "imagefile",
     "source" :"4"
}
```

- **mask= select an image mask produced by another node. Diff masks are specific to an edge: a source and target node pair. An edge is identified by source and target node ids. The mask is used by some plugins to identify areas to adjust.**

```
 {
  "type" : “mask",
  "source" :"4"
  “target" :“6“
}
```

- **value=provide a specific value**

```
"Image Rotated" : {
  "type" : "value",
  "value" :"yes"
}
```

- **donor=pick resulting image from a predecessor node. ‘source‘ is optional.**

```
”input" : {
  "type" : ”donor",
  “source”:”3”
}
```

- **list=pick from a set of values**

```
”subject" : {
  "type" : ”list",
  "values" :[”other”, “landscape”]
}
```

Argument Types Cont.

- **variable = select an output name/value pair from a predecessor plugin node. The source node is provided using it’s node id. The pair’s value is identified by the name**

```
“box_to_alter “: {
      "type" : “variable",
      “name" : “box_altered“,
        "source" :"4"
 }
```

- **input= identifies the name of input image file (input mask) from the output (target) image of another plugin.**

```
 “inputmask“ : {
   "type" : “input",
   “source" :“6“
 }
```

- **plugin=call a plugin function registered through the Python setuptool‘s entry point *maskgen_specs*. The function name is the entry point name. The function accepts a dictionary of parameters, provided in the definition.**

```
”kernel" : {
  "type" : ”plugin”,
 “name" : “kernel_builder“,
  ”parameters" : { “kernel_size”: 5}
```

- **Int[low:high]=pick a value, uniform distribution over the range, inclusive.**

```
”kernal_size" : {
  "type" : ”int[1:100]”
}
```

- **float[low:high]=pick a value, gaussian distribution over the range, inclusive**

```
”percent_change" : {
  "type" : ”float[0.0:1.0]”
}
```

- **yesyno=pick yes or no**

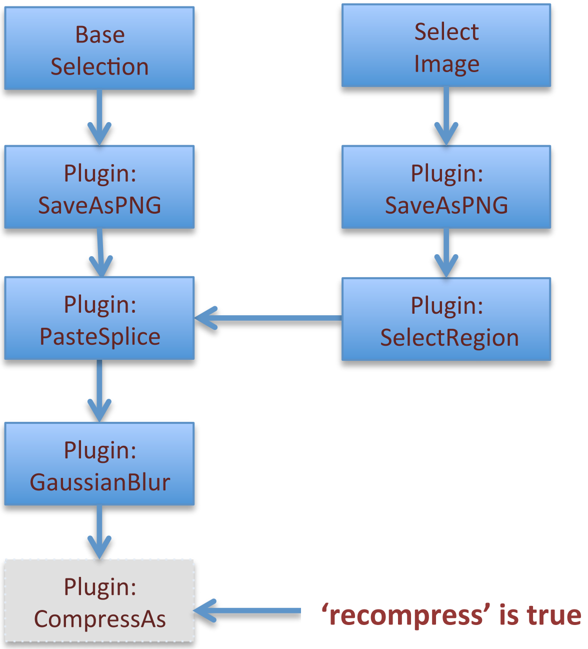
```
”color_correct" : {
  "type" : ”yesno”
}
```

Example JSON

```
{
 "directed": true,
 "graph": {
  "username": "ericrobertson",
  "name": "sample",
  "organization": "PAR"
 },
 "nodes": [
  {
   "op_type": "BaseSelection",
   "image_directory": "tests/images",
   "picklist": "imageset",
   "id": "0"
  },
  {
   "op_type": "ImageSelection",
   "image_directory": "tests/images",
   "picklist": "imageset",
   "id": "1"
  },
  {
   "op_type": "PluginOperation",
   "plugin": "SaveAsPNG",
   "picklist": "imageset",
   "id": "2",
   "arguments": {
    "Image Rotated" : {
     "type" : "value",
     "value" :"yes"
    }
   }
  },
  {
   "op_type": "PluginOperation",
   "plugin": "SaveAsPNG",
   "picklist": "imageset",
   "id": "3",
   "arguments": {
    "Image Rotated" : {
     "type" : "value",
     "value" :"yes"
    }
   }
  },
  {
   "op_type": "PluginOperation",
   "plugin": "SelectRegion",
   "id": "4",
   "arguments": {}
  },
  {
   "op_type": "PluginOperation",
   "plugin": "PasteSplice",
   "id": "5",
   "arguments": {
    ”donor" : {
     "type" : ”donor",
    }
   }
  },
  {
   "op_type": "PluginOperation",
   "plugin": "GaussianBlur",
   "id": "6",
   "arguments": {
    "inputmaskname" : {
     "type" : "mask",
     "source" :"2",
     "target": "5”
    }
   }
  }
 ],
 "links": [
  {
   "source": 0,
   "target": 2
  },
  {
   "source": 1,
   "target": 3
  },
  {
   "source": 3,
   "target": 4
  },
  {
   "source": 2,
   "target": 5
  },
  {
   "source": 4,
   "target": 5
  },
  {
   "source": 5,
   "target": 6
  }
 ],
 "multigraph": false
}
```

Graph View

Process Graph



Base Selection

Plugin: SaveAsPNG

Plugin: PasteSplice

Plugin: GaussianBlur

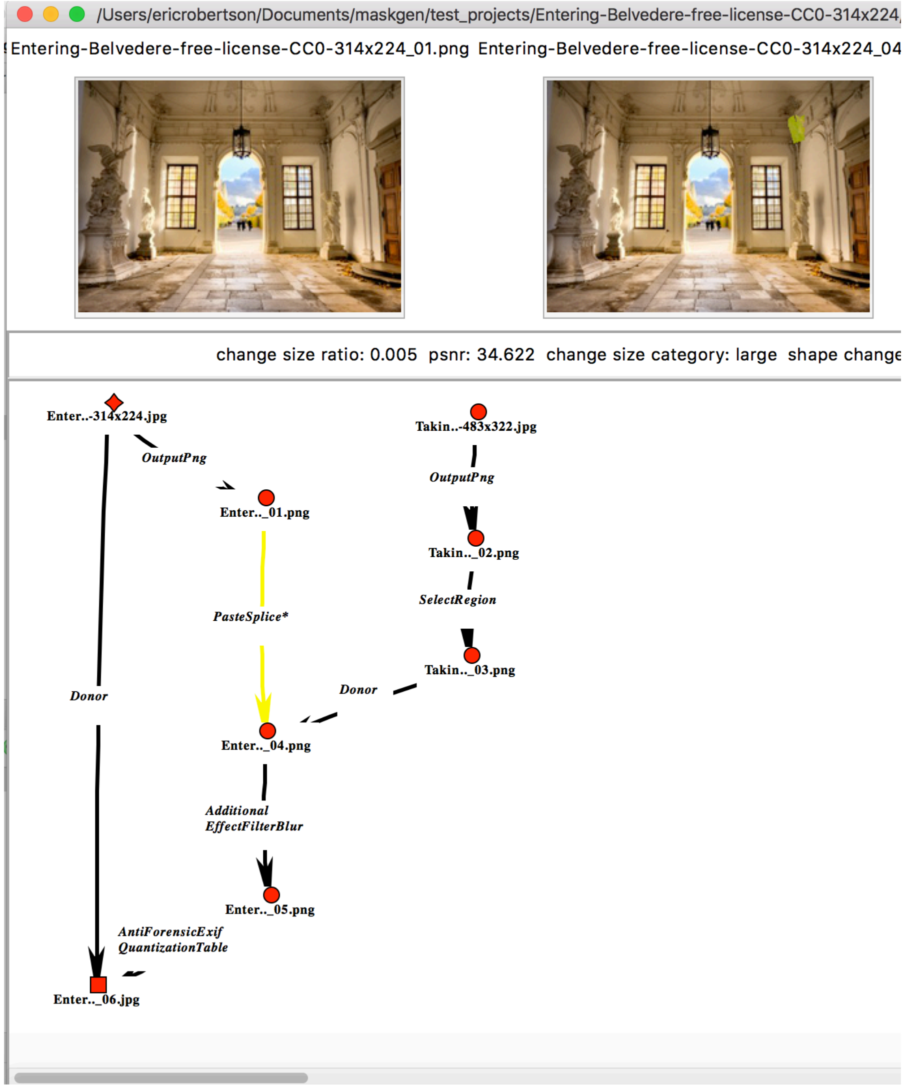
Plugin: CompressAs

Select Image

Plugin: SaveAsPNG

Plugin: SelectRegion

'recompress' is true



/Users/ericrobertson/Documents/maskgen/test_projects/Entering-Belvedere-free-license-CC0-314x224/

Entering-Belvedere-free-license-CC0-314x224_01.png Entering-Belvedere-free-license-CC0-314x224_04

change size ratio: 0.005 psnr: 34.622 change size category: large shape change

Running

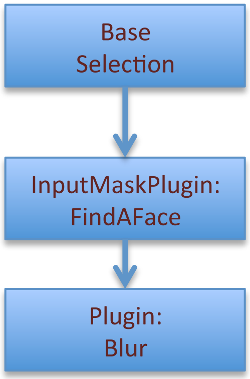
- Install JT
- Make sure resources/*.json are moved to where the tool is being executed, or run the tool from the JT *maskgen* directory.
- Example Command Line:

```
python maskgen/batch/batch_project.py --count 2 --results tests/test_projects
 --json tests/batch_process.json --loglevel 0
```

- Arguments
  - Mandatory
    - results = directory to hold completed projects
    - json = the batch process JSON description file
  - Optional
    - count = number of projects to build.
      - Make sure count < number of images in the select image directory
      - By default, just creates one.
      - A value of 0 is used with the ‘graph’ option.
    - loglevel = 0 to 50 log level, 0 being finest
    - graph = create a Graphviz layout. File name is the name of the batch process + ‘.png’.
    - threads = number of threads to run, building project in parallel (one thread per project). Default is 1.

Using the InputMaskPlugin

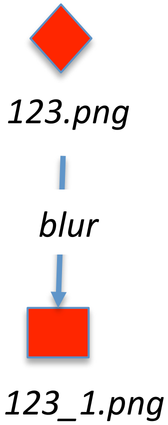
Process Graph



Enable pre-selection of pixels to manipulate by succeeding operation nodes.

In this example, identify regions to blur using a plugin that identifies faces.

Journal Graph



Installing Plugin Functions

Plugin functions are used to set parameters for operations.

Create separate project and install it with the user-defined function. The key piece is the following in setup.py:

```
    entry_points=
            {'maskgen_specs': [
                 ’foo = myplugin.foo:getLength'
             ]
            },
```

Install the project (*python setup.py install*)

The name *foo* is used in as the item name of the plugin type argument specification.

This is a pseudo-name for the function getLength() in the package *myplugin.foo*. The entry point name maskgen_specs is a locator for the JT batch to discover these plug-in specification functions.

Example Operation Node Definition:

```
    "id": "MaskSelect",
    "op_type": "InputMaskPluginOperation",
    "arguments": {
       "percentage_width": {
       "type": "plugin",
       "name" : "foo",
       "parameters" : {"param1" : "whatever"}
      }
```