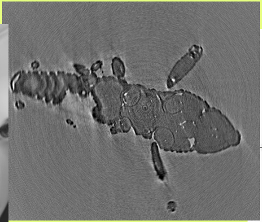
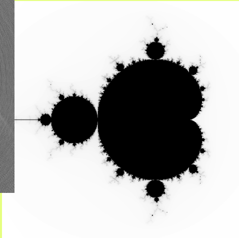
Updates on *Fast Reject* and the *Computing Framework*

Matthias Vogelgesang, Michele Caselle

Institut für Prozessdatenverarbeitung und Elektronik



Fast Reject Review

```
n_r ← 0
for row r_{k−1} ∈ I_{k−1}, r_k ∈ I_k do
   n_p ← 0
   for pixel p ∈ ‖r_{k−1} − r_k‖ do
      if p ≥ t_p then
         n_p ← n_p + 1
      end if
   end for
   if n_p ≥ t_r then
      n_r ← n_r + 1
   end if
end for
if n_r < t_I then
   Reject frame
end if
```

- I_k: Image at time k
- n_r: Number of triggered rows
- n_p: Number of triggered pixels per row
- t_p: Pixel threshold (luminance)
- t_r: Pixel threshold per row (number)
- t_I: Row threshold per image (number)

Pearson Correlation Coefficient

- Two subsequent frames can be thought of as realizations of a stochastical process in time
- Let X and Y denote random variables associated with these images
- We are interested in how similar the images are, respectively how those random variables *correlate*
- Correlation coefficient

$$\rho_{X,Y} = \frac{\mathrm{cov}(X, Y)}{\sigma_X \sigma_Y}$$

- Thus, the sample correlation coefficient

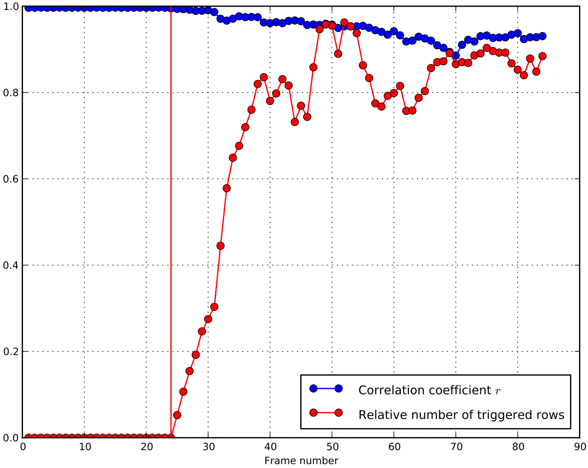
$$r(X, Y) = \frac{\sum_i^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_i^n (X_i - \bar{X})^2}\sqrt{\sum_i^n (Y_i - \bar{Y})^2}}$$

Correlation Coefficient as Image Similarity Measure

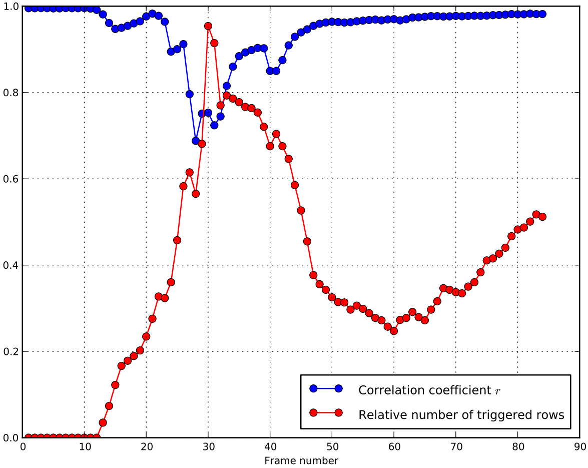
Properties of the Coefficient

- $-1 \leq r \leq 1$ describes linear association between X and Y
- $r = 1 \Rightarrow$ images are identical
- $r(a + bX, c + dY) = r(X, Y) \Rightarrow$ change in brightness does not affect r

Results for ‘screws’



Results for ‘trap’



Interpretation

- Visible examination tells that $r < 0.98$ implies small but significant change
- On the other hand, fast reject triggers as soon as r drops below that value

Problems

- Is this really a suitable tool?
  - Small global changes (scale, transform, rotation) affect r negatively
  - Wrong premise: Proving correctness with a number that is generated in a similar way to how the algorithm works
- “Fast” Reject Algorithm subsamples rows, thus decreasing vertical resolution
- Parameter set (t_p, t_r, t_l) depends on application and must be determined *a priori*

General changes to the Framework

- Re-design of the structure
- New filter nodes
  - Elena's NLM noise reduction
  - Complex arithmetics
  - Multiplexer/Demultiplexer
  - Direct OpenCL
- Integrate into Jenkins CI

Change of Structure

Requirements

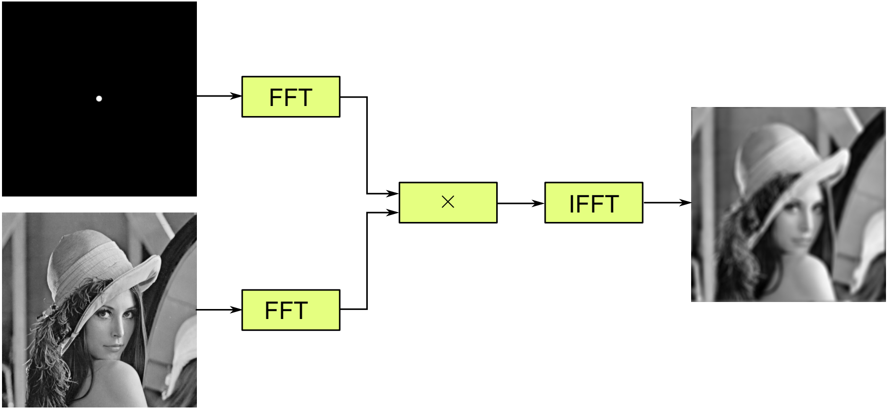
- Some algorithms demand several in- or outputs
- Better scheduler needed a bit more flexible structure
- Keep JSON description format

Changes

- Nodes are connected directly using `UfoChannels`
- Multiple inputs or outputs are named and bound via that name
  - Simpler filters use `default` input and output channels
- Split structure is obsolete and can be emulated by
  - connecting to several inputs
  - connecting from several outputs
  - using the `demux` node

What does that mean?

Let’s pretend we want to do a fast 2D convolution with a large kernel using our precious GPUs. . .



Initialization

Instead of JSON, we will use Python here:

```
from gi.repository import Ufo

g = Ufo.Graph()

# instantiate plugins
lena = g.get_filter('reader')
kernel = g.get_filter('reader')
fft1 = g.get_filter('fft')
fft2 = g.get_filter('fft')
complx = g.get_filter('complex')
ifft = g.get_filter('ifft')
writer = g.get_filter('writer')
fftshift = g.get_filter('cl')
```

Initialization

Instead of JSON, we will use Python here:

```
from gi.repository import Ufo

g = Ufo.Graph()

# instantiate plugins
lena = g.get_filter('reader')
kernel = g.get_filter('reader')
fft1 = g.get_filter('fft')
fft2 = g.get_filter('fft')
complx = g.get_filter('complex')
ifft = g.get_filter('ifft')
writer = g.get_filter('writer')
fftshift = g.get_filter('cl')
```

Configuring Filters

```
# configure nodes
lena.set_properties(path='/home/matthias/data/lena/',
    prefix='lena')
kernel.set_properties(path='/home/matthias/data/lena/',
    prefix='kernel')

fft1.set_properties(dimensions=2)
fft2.set_properties(dimensions=2)
ifft.set_properties(dimensions=2)
complx.set_properties(operation='mul')
fftshift.set_properties(file='kernels.cl',
    kernel='fftshift', inplace=False)
```

Connecting Filters


```
# connect the nodes
lena.connect_to(fft1)
kernel.connect_to(fft2)

fft1.connect_by_name('default', complx, 'input1')
fft2.connect_by_name('default', complx, 'input2')

complx.connect_to(ifft)
ifft.connect_to(fftshift)
fftshift.connect_to(writer)

# defocus Lena
g.run()
```

Connecting Filters

```
# connect the nodes
lena.connect_to(fft1)
kernel.connect_to(fft2)

fft1.connect_by_name('default', complx, 'input1')
fft2.connect_by_name('default', complx, 'input2')

complx.connect_to(ifft)
ifft.connect_to(fftshift)
fftshift.connect_to(writer)

# defocus Lena
g.run()
```

Connecting Filters

```
# connect the nodes
lena.connect_to(fft1)
kernel.connect_to(fft2)

fft1.connect_by_name('default', complx, 'input1')
fft2.connect_by_name('default', complx, 'input2')

complx.connect_to(ifft)
ifft.connect_to(fftshift)
fftshift.connect_to(writer)

# defocus Lena
g.run()
```

Connecting Filters

```
# connect the nodes
lena.connect_to(fft1)
kernel.connect_to(fft2)

fft1.connect_by_name('default', complx, 'input1')
fft2.connect_by_name('default', complx, 'input2')

complx.connect_to(ifft)
ifft.connect_to(fftshift)
fftshift.connect_to(writer)

# defocus Lena
g.run()
```

Any Questions?

?