

Machine vision and device integration with the Ruby Programming Language

J. Wedekind^a

Friday, February 29th 2008

Microsystems and Machine Vision Laboratory
Modelling Research Centre, Materials Engineering Research Institute



^aDr J. R. Travis, Dr M. Thompson, Dr B. P. Amavasai, Nanorobotics EPSRC Basic Technology grant [GR/S85696/01](#)

Introduction Nanorobotics Project

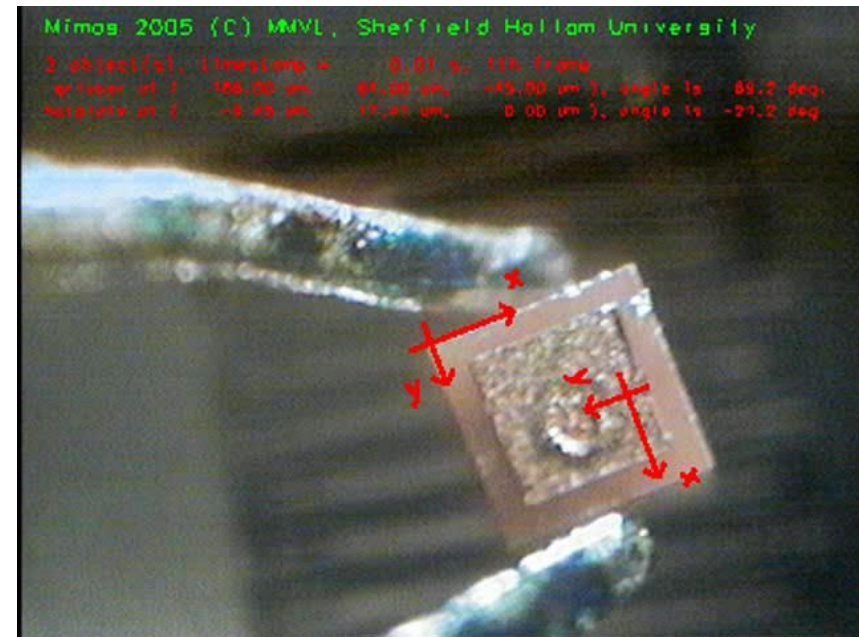
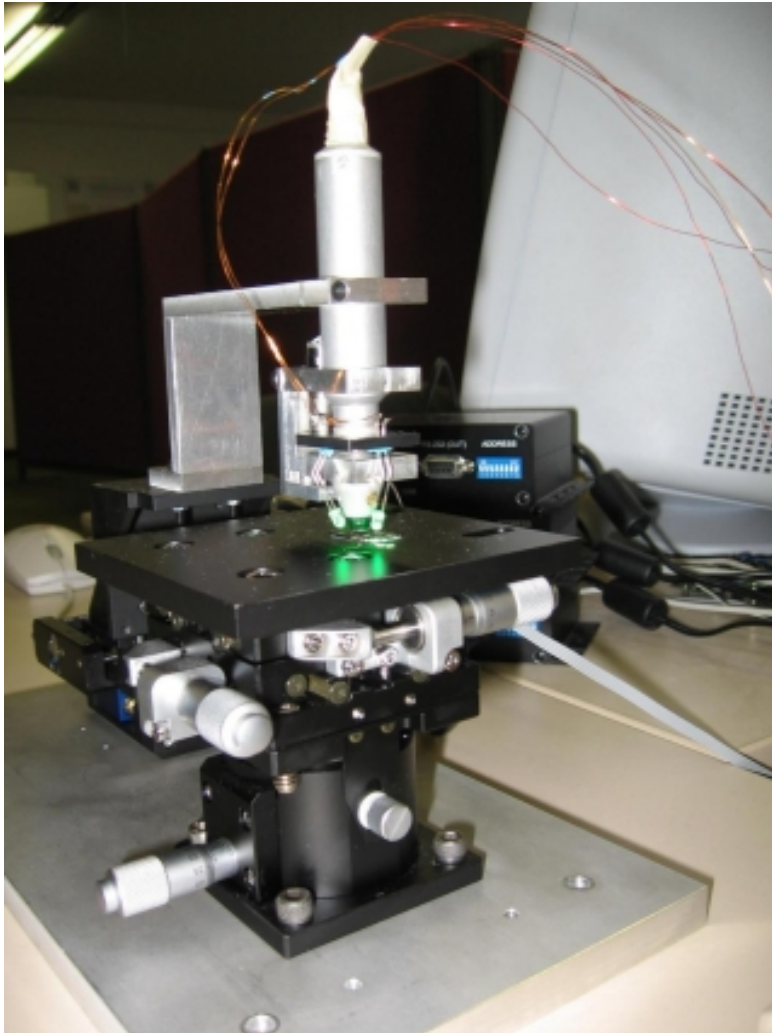
environment

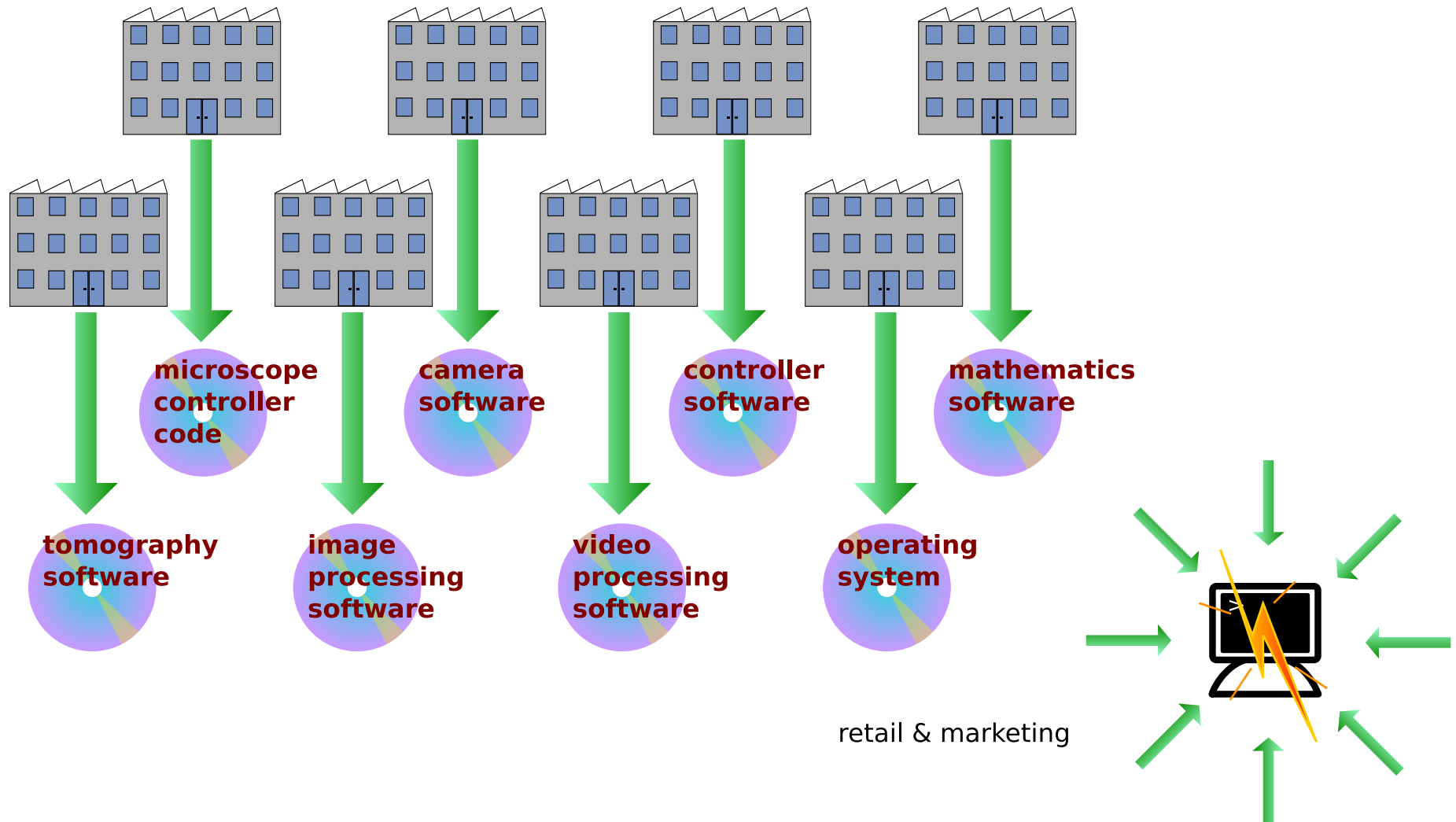
- transmission electron microscope
- digital camera
- piezo controller
- nano indenter



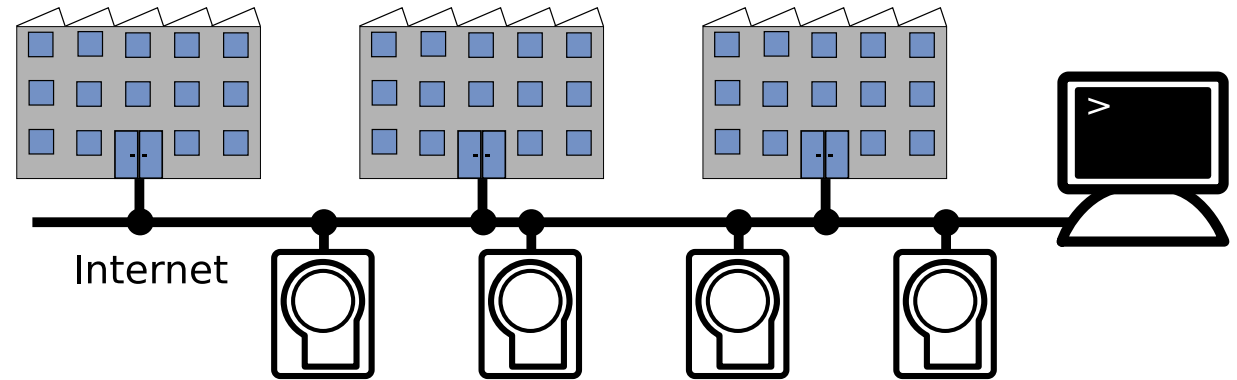
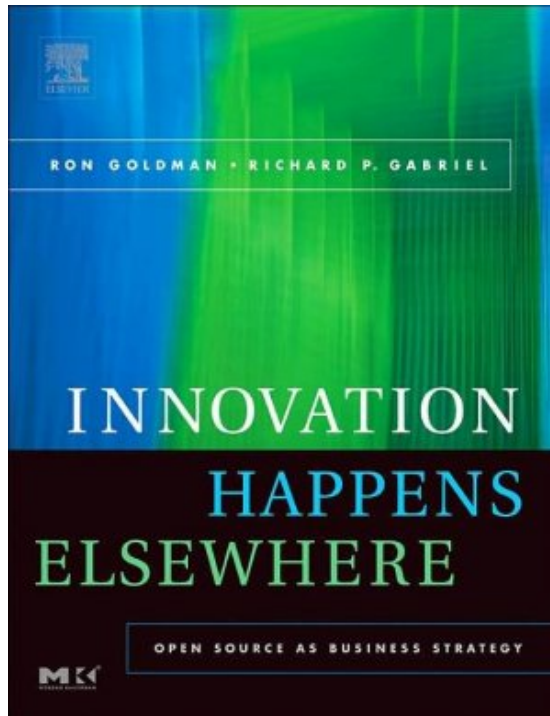
environment

- micro camera
- digital camera
- piezo drives
- gripper





Introduction Community Development Model



“The market need is greatest for platform products because of the importance of a reliable promise that vendor lock-in will not endanger the survival of products built or modified on the software stack above that platform.” - Ron Goldman & Richard P. Gabriel

“It is important to remove as many barriers to collaboration as possible: social, political, and technical.” - Ron Goldman & Richard P. Gabriel





four freedoms (Richard Stallman)

1. The freedom to run the program, for any purpose.
2. The freedom to study how the program works, and adapt it to your needs.
3. The freedom to redistribute copies so you can help your neighbor.
4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits.



respect the freedom of downstream users (Richard Stallman)

GPL requires derived works to be available under the same license.

covenant not to assert patent claims (Eben Moglen)

GPLv3 deters users of the program from instituting patent litigation by the threat of withdrawing further rights to use the program.

other (Eben Moglen)

GPLv3 has regulations against DMCA restrictions and tivoization.



**relative size
of binaries**

OpenEXR



Ruby/IRB



HornetsEye

NArray



Qt4/
QtRuby4



ImageMagick/
RMagick



Ruby Programming Language Tiobe Index

| Position Feb 2008 | Position Feb 2007 | Delta in Position | Programming Language | Ratings Feb 2008 | Delta Feb 2007 | Status |
|-------------------|-------------------|-------------------|----------------------|------------------|----------------|--------|
| 1 | 1 | = | Java | 21.483% | +2.50% | A |
| 2 | 2 | = | C | 14.859% | -1.24% | A |
| 3 | 5 | ↑↑ | (Visual) Basic | 11.604% | +3.24% | A |
| 4 | 4 | = | PHP | 9.890% | +1.04% | A |
| 5 | 3 | ↓↓ | C++ | 9.274% | -1.49% | A |
| 6 | 6 | = | Perl | 6.205% | +0.13% | A |
| 7 | 7 | = | Python | 4.763% | +1.20% | A |
| 8 | 8 | = | C# | 4.510% | +1.32% | A |
| 9 | 12 | ↑↑↑ | Delphi | 2.798% | +0.72% | A |
| 10 | 9 | ↓ | JavaScript | 2.334% | -0.65% | A |
| 11 | 10 | ↓ | Ruby | 1.862% | -0.67% | A |
| 12 | 15 | ↑↑↑ | D | 1.190% | -0.01% | A |
| 13 | 13 | = | PL/SQL | 0.981% | -0.65% | A |
| 14 | 11 | ↓↓↓ | SAS | 0.949% | -1.38% | A |
| 15 | 18 | ↑↑↑ | COBOL | 0.842% | +0.19% | A |
| 16 | 22 | ↑↑↑↑↑ | FoxPro/xBase | 0.538% | +0.02% | B |
| 17 | 19 | ↑↑ | Pascal | 0.445% | -0.15% | B |
| 18 | 44 | ↑↑↑↑↑↑↑↑ | Lua | 0.388% | +0.27% | B |
| 19 | 17 | ↓↓ | Ada | 0.385% | -0.28% | B |
| 20 | 16 | ↓↓↓ | Lisp/Scheme | 0.354% | -0.37% | B |

<http://www.tiobe.com/>

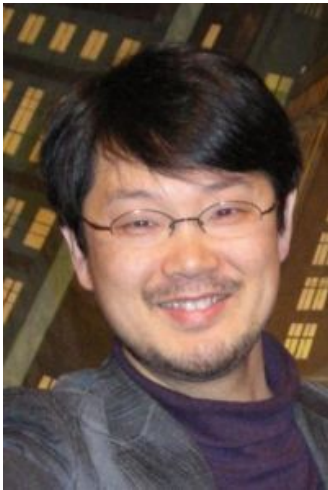


Ruby Programming Language Speed Comparison

| Language | Implementation | Time | Relative Speed |
|----------|----------------|---------|----------------|
| C | gcc 4.0.1 | 0.05 s | 1.00× |
| Java | Java 1.4.2 | 0.40 s | 8.00× |
| Lua | Lua 5.1 | 1.50 s | 30.00× |
| Python | Python 2.5.1 | 9.99 s | 199.80× |
| Perl | Perl 5.8.6 | 21.75 s | 435.00× |
| Ruby | Ruby 1.8.4 | 34.31 s | 686.18× |
| Lisp | Emacs Lisp | 47.25 s | 945.00× |

<http://www.timestretch.com/FractalBenchmark.html>





Ruby

- created by Yukihiro Matsumoto
- released 1995
- inspired by Perl, Python, Smalltalk, Eiffel, Ada, and Lisp
- “pseudo simplicity”: simple syntax \Leftrightarrow multi-paradigm language
- highly portable

“Ruby is simple in appearance, but is very complex inside, just like our human body.” - Yukihiro Matsumoto

Ruby on Rails

- web application framework based on Ruby
- created by David Heinemeier Hansson
- released 2004

- interpreted language (“def” is a statement which defines a method)
- dynamically typed variable: name, value
- everything is an object (no basic types as in C++, Java, ...)
- type inspection



- mark-and-sweep garbage collector
- object orientation, dynamic single-dispatch
- reflection (message passing)
- metaprogramming (overloading methods during runtime)



- graph of context objects (using garbage collector) instead of global namespace and stack
- closures (lambda expression with context)
 - unifying concept for function objects, iterators
 - control structures can be exported as methods
- continuations instead of goto/break
- mixins: unifying concept for namespaces and interfaces
- exception handling: decouple occurrence and handling of error





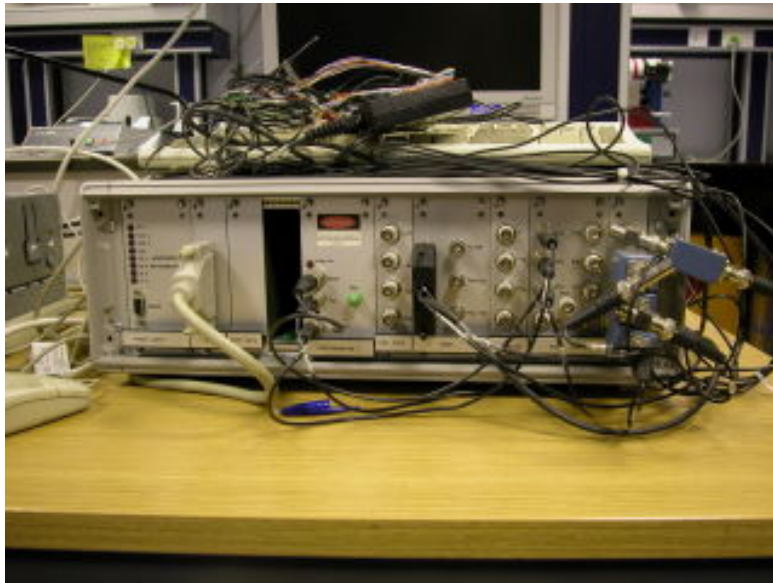
mbrot.cc

```
// g++ -shared -fPIC -I/usr/lib/ruby/1.8/x86_64-linux -o mbrot.so mbrot.cc
#include <complex>
#include <ruby.h>
VALUE iterate( VALUE rbSelf, VALUE rbX, VALUE rbY )
{
    std::complex< double > c( NUM2DBL( rbX ), NUM2DBL( rbY ) ), z( 0, 0 );
    for ( int i=0; i<100; i++ ) {
        z = z * z + c;
        if ( std::norm( z ) > 4 ) return rb_str_new2( " " );
    };
    return rb_str_new2( "#" );
}
extern "C" void Init_mbrot(void) {
    VALUE cMandelbrot = rb_define_class( "Mandelbrot", rb_cObject );
    rb_define_method( cMandelbrot, "iterate", RUBY_METHOD_FUNC( iterate ), 2 );
    rb_require( "mbrot_ext.rb" );
}
```





Serial Interface, ECL



Slider

North

Up

West

East

South

Down

Mode

Pressed

Continuous

Pulse 10

Coordinates

x = 0 y = 0 z = 0 Reset

Button

West 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

```

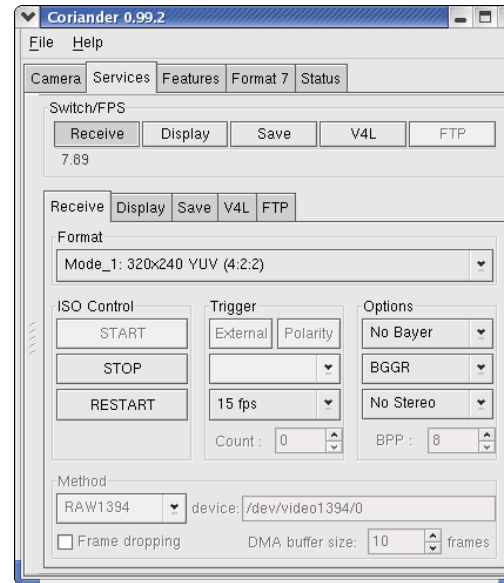
l=2000
s=2
v=0.7
a=65535*v
(0...l).collect do |x|
  a * ( Math::exp( s * x.to_f/l ) - 1 ) / ( Math::exp(s) - 1 )
end + [0]

```

Evaluate
Quit

PCI-DIO24 driver by Warren Jasper

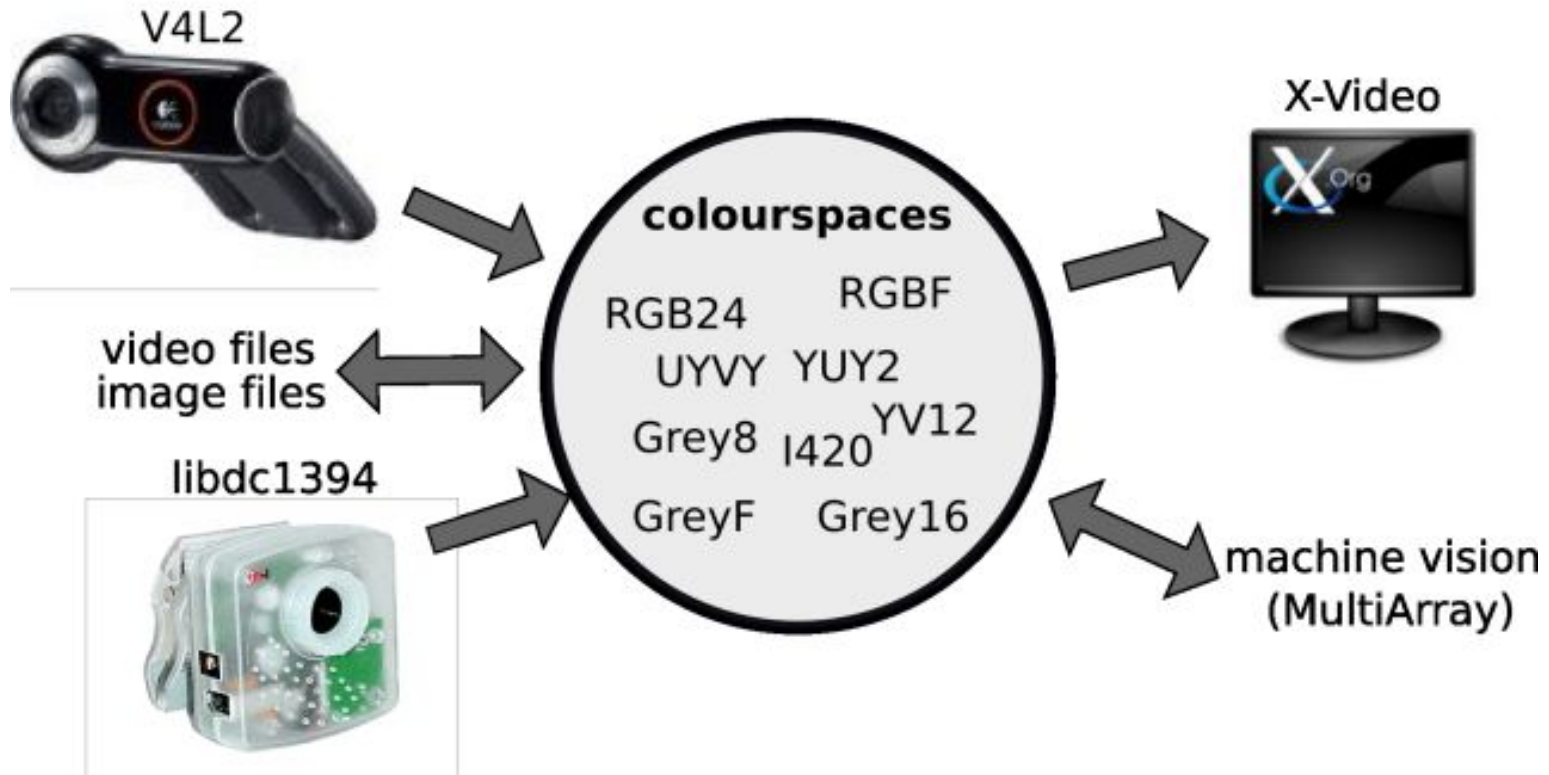
Device Integration TVIPS Firewire Digital Camera



- IEEE1394 electronic interface
- IIDC/DCAM protocol

libdc1394 and coriander by
Damien Douchamps

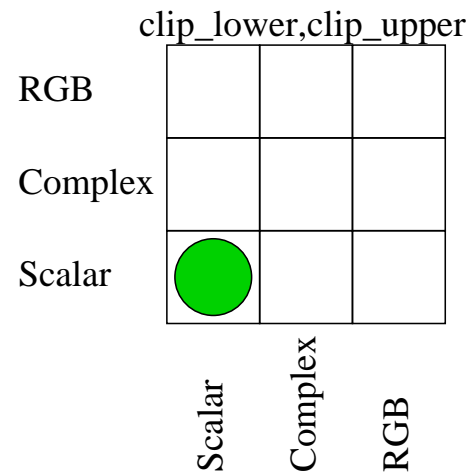
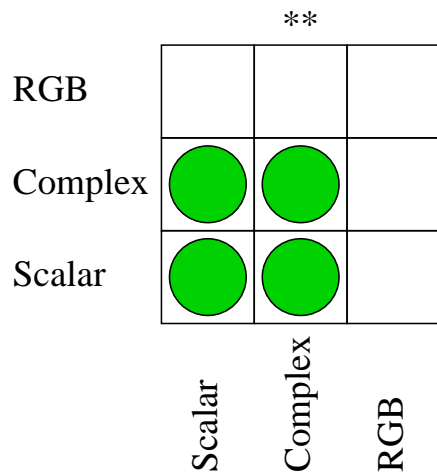
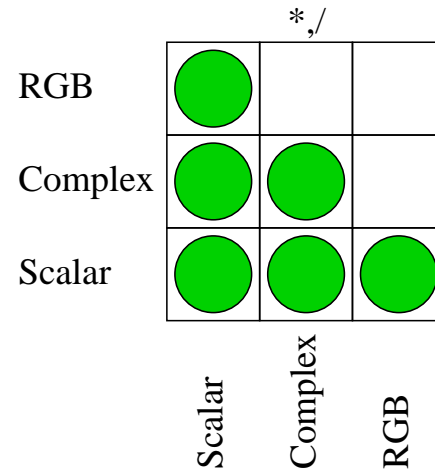
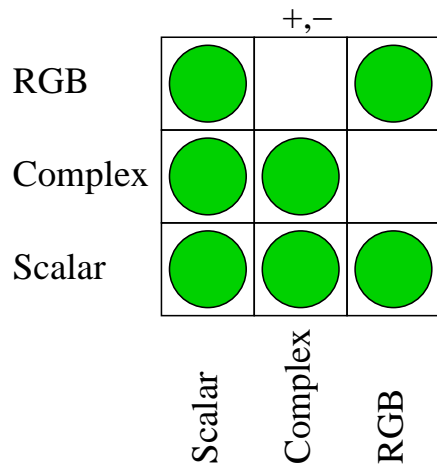




$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.500 \\ 0.500 & -0.418688 & -0.081312 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix}$$

also see: <http://fourcc.org/>



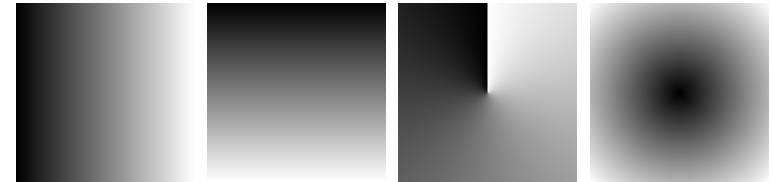


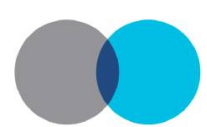
Computing "img/2"

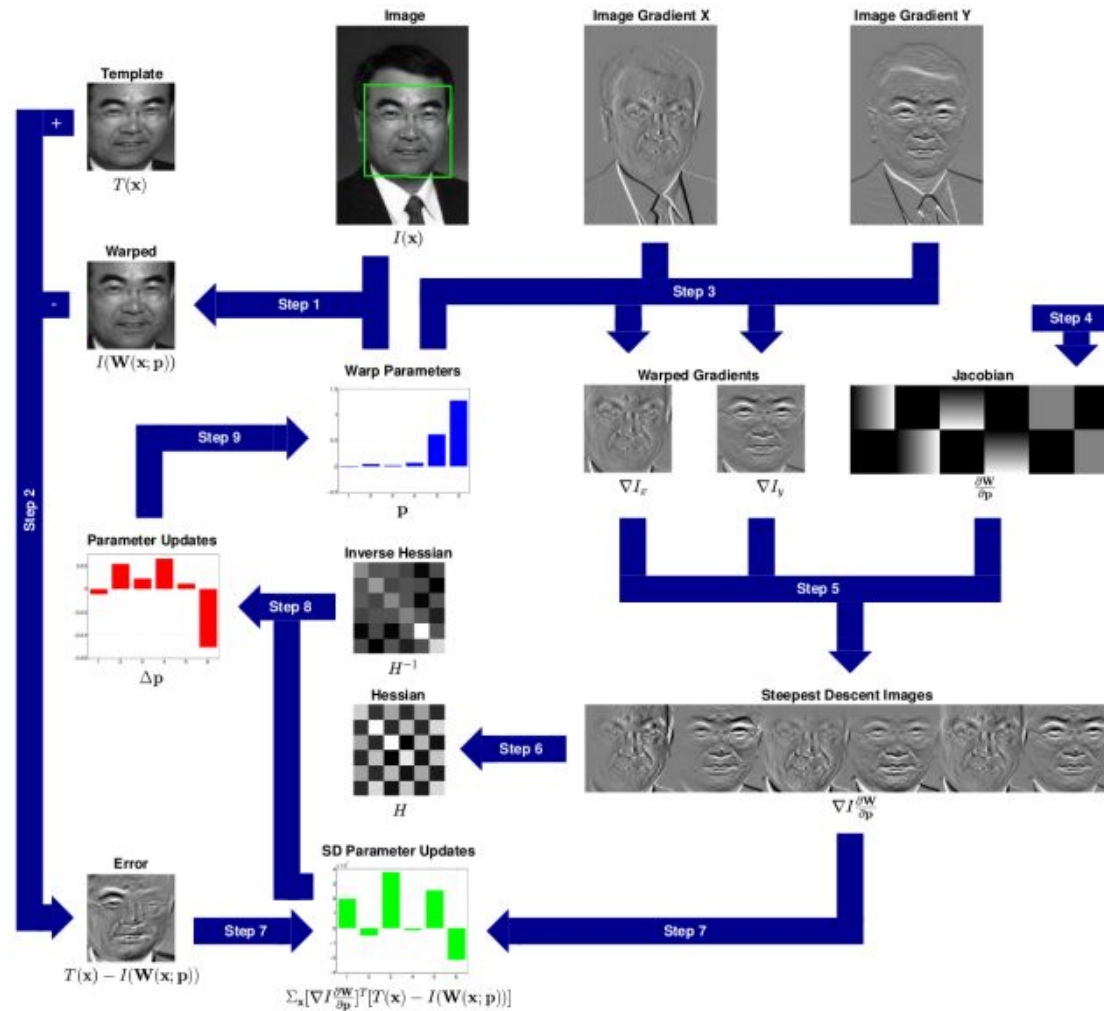
1. Message "/" with parameter "2" is send to "img"
2. Message is dispatched to class "MultiArray"
3. Method "scalarright_div_ubyte_ubyte_rgb" is checked for
4. Native implementation is invoked

```
w, h = img.shape[0], img.shape[1] / 2
v = MultiArray.new( MultiArray::LINT,
                    h, h, 2 )

x = xramp( h, h )
y = yramp( h, h )
c = 0.5 * h
v[ 0...h, 0...h, 0 ] =
  ( ( ( x - c ).atan2( y - c ) / PI + 1 ) *
    w / 2 - 0.5 )
v[ 0...h, 0...h, 1 ] =
  ( ( x - c ) ** 2 + ( y - c ) ** 2 ).sqrt
result = img.warp_clipped( v )
```







http://www.ri.cmu.edu/projects/project_515.html



Initialisation

```
p = Vector[ xshift, yshift, rotation ]
x, y = xramp( *tpl.shape ), yramp( *tpl.shape )
gx = tpl.gauss_gradient_x( sigma )
gy = tpl.gauss_gradient_y( sigma )
c = Matrix[ [ 1, 0 ], [ 0, 1 ], [ -y, x ] ] * Vector[ gx, gy ]
hs = ( c * c.covector ).collect { |e| e.sum }
```

Tracking Iteration

```
field = MultiArray.new( MultiArray::LINT, w, h, 2 )
field[ 0...w, 0...h, 0 ] = x * cos( p[2] ) - y * sin( p[2] ) + p[0]
field[ 0...w, 0...h, 1 ] = x * sin( p[2] ) + y * cos( p[2] ) + p[1]
diff = img.warp_clipped( field ).to_type( MultiArray::SFLOAT ) - tpl
s = c.collect { |e| ( e * diff ).sum }
d = hs.inverse * s
p += Matrix[ [ cos(p[2]), -sin(p[2]), 0 ],
             [ sin(p[2]),  cos(p[2]), 0 ],
             [ 0,          0, 1 ] ] * d
```





conclusion

- native implementation in C++, Ruby as glue-code
- development platform for microscopy software and general purpose machine vision
- used for medical imaging, industrial inspection, measuring abrasion

future work

- feature-based object recognition algorithm
- GPU acceleration, parallel processing
- support typical workflow on a microscope

web

<http://www.wedesoft.demon.co.uk/hornetseye-api/>

<http://rubyforge.org/projects/hornetseye/>

<http://sourceforge.net/projects/hornetseye/>

