

SU2-6: Video Tracking

SU2-6(4): A Machine Vision Extension for the Ruby Programming Language

J. Wedekind, B. P. Amavasai, K. Dutton, M. Boissenin

Sunday, June 22th 2008

Microsystems and Machine Vision Laboratory
Materials Engineering Research Institute
Sheffield Hallam University
Sheffield
United Kingdom



Introduction

EPSRC Nanorobotics grant



Slider

Slider Piezos

Fine Piezos

Coordinates

Name	x	y	z
1 a	0.3	0.3	20.0
2 b	0.2	0.25	18.0

name = b Add Entry

x = 0,2000 V 0.00691200018055838

y = 0,2500 V 0.00864000022569798

z = 18,0000 V 19.9308799981944

x-y-speed = 0,1000 V/s

z-speed = 0,1000 V/s

x-y-incr. = 0,1000 V

z-incr. = 0,1000V

pulsewidth = 0.1

delay = 0.0

length = (@fineFrequency * pulsewidth)to_j

length2 = length / 2


scale = 1.0

ramp = (0...length2).collect { |x| scale * x.to_f / (length2 - 1) }

iramp = ramp.collect { |x| scale - x }

retval = ["a", "b", ramp + iramp + [0] * (@fineFrequency * delay).to_j]

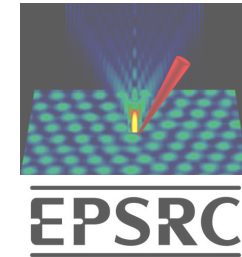
retval



Evaluate Run 100 times Calibrate 10767.4999429657 Hz

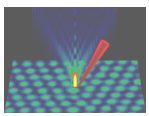
Log

Quit



electron microscopy

- telemanipulation
- drift-compensation
- closed-loop control
- computer vision**
- real-time software
- system integration
- theoretical insights



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)

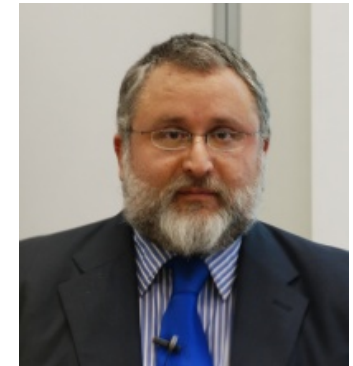
GPL requires derived works to be available under the same license.

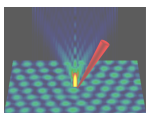
covenant not to assert patent claims (Eben Moglen)

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.





The screenshot shows the Ruby website homepage. At the top left is the Ruby logo, a red gemstone, with the text "Ruby" and the tagline "A Programmer's Best Friend". To the right is a search bar with a "Search" button. Below the header is a navigation menu with links for "Downloads", "Documentation", "Libraries", "Community", "News", and "About Ruby". The main content area is divided into three columns. The left column is titled "Ruby is..." and describes Ruby as a dynamic, open source programming language with a focus on simplicity and productivity. The middle column contains a code block with three examples of Ruby code and their outputs. The right column has three sections: "Download Ruby" with a download icon, "Get Started, it's easy!" with links for "Try Ruby! (in your browser)", "Ruby in Twenty Minutes", and "Ruby from Other Languages", and "Explore a new world..." with links for "Documentation" and "Books".

Ruby is...

A dynamic, open source programming language with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write.

[Read More...](#)

```
# Output "I love Ruby"
say = "I love Ruby"
puts say

# Output "I *LOVE* RUBY"
say[ 'love' ] = "*love*"
puts say.upcase

# Output "I *love* Ruby"
# five times
5.times { puts say }
```

[Download Ruby](#)

Get Started, it's easy!

[Try Ruby! \(in your browser\)](#)

[Ruby in Twenty Minutes](#)

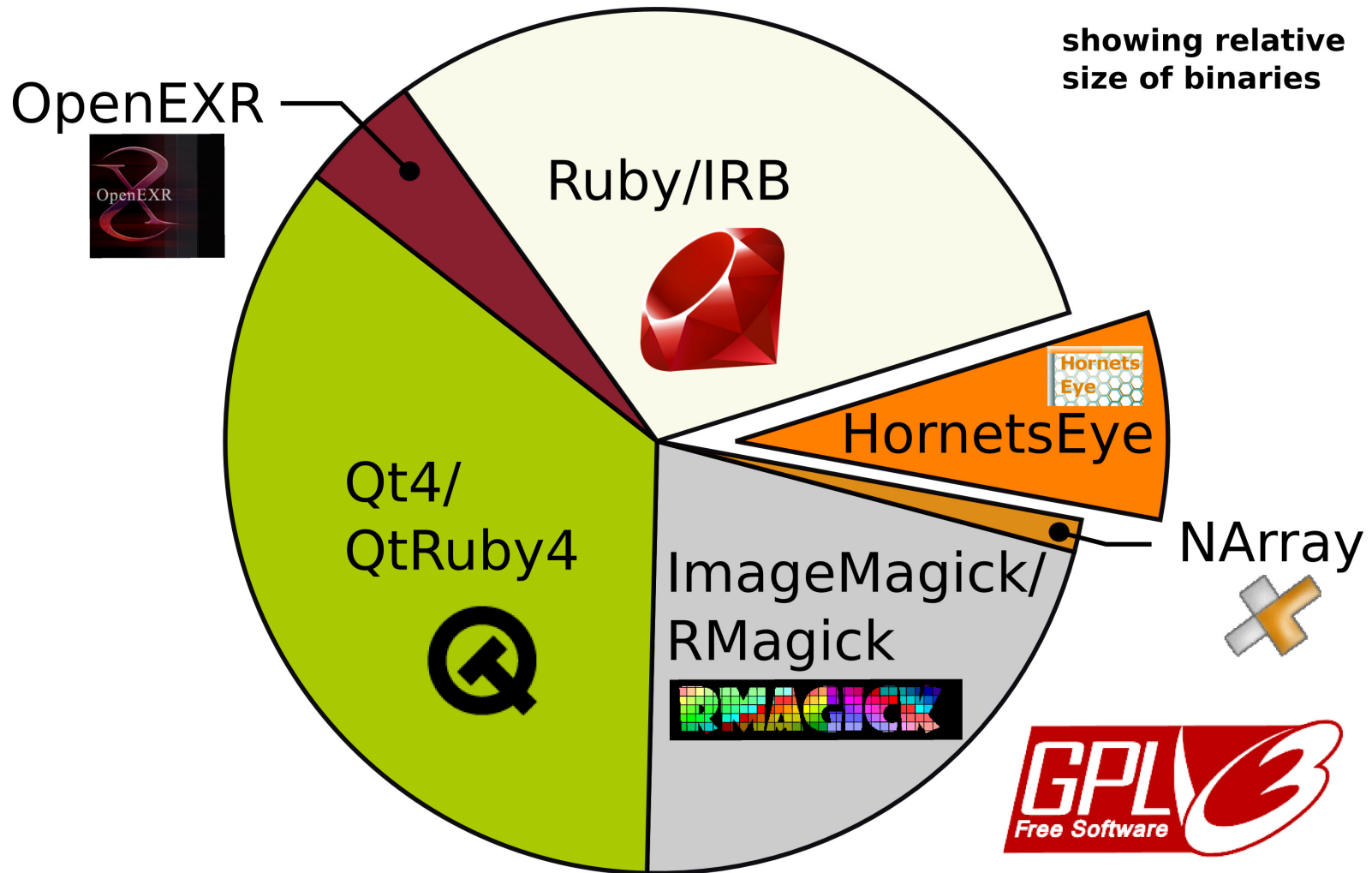
[Ruby from Other Languages](#)

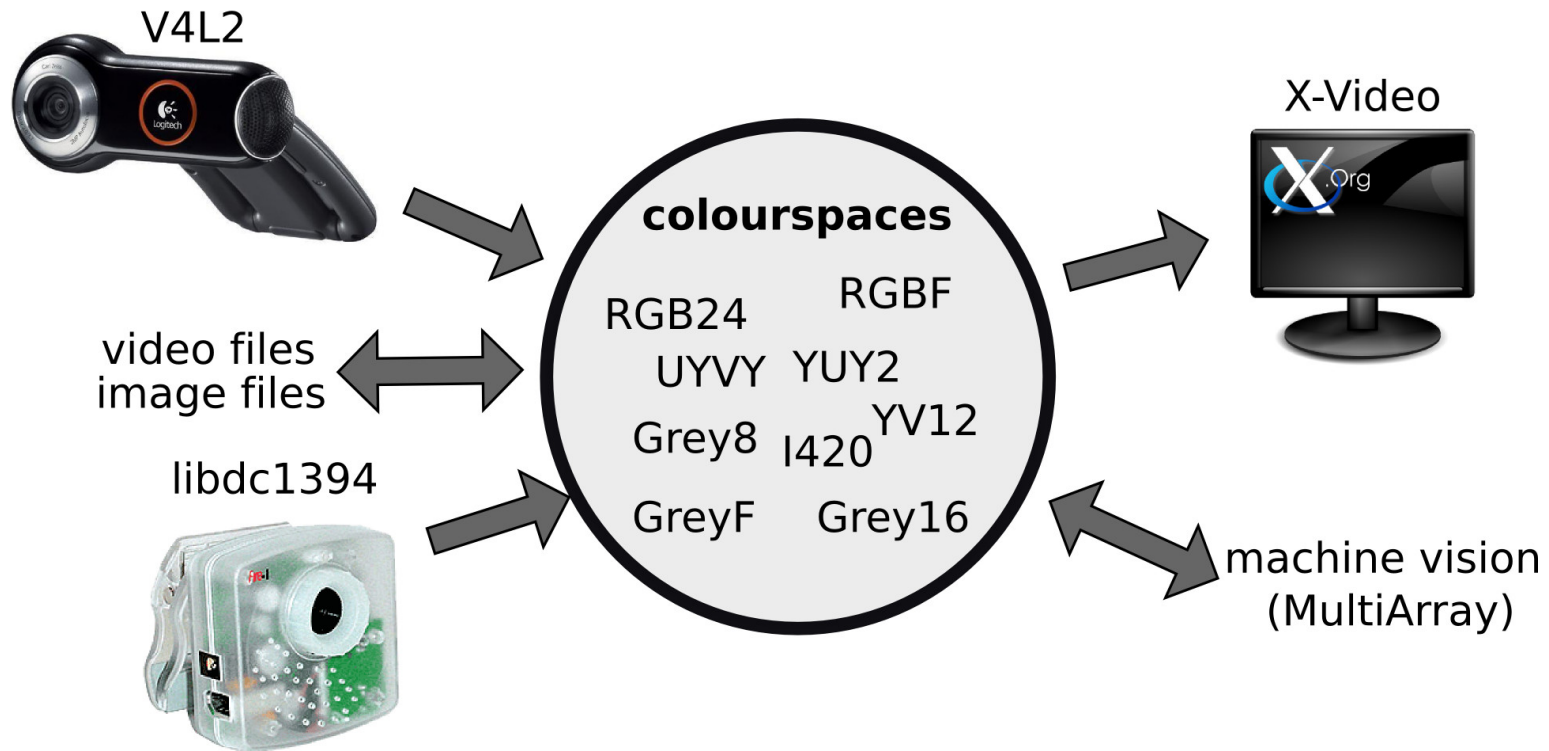
Explore a new world...

[Documentation](#)

[Books](#)

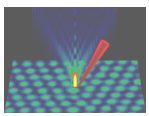
<http://www.ruby-lang.org/>





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

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



HornetsEye

Multi-dimensional arrays

$$g, h \in \{0, 1, \dots, w - 1\} \times \{0, 1, \dots, h - 1\} \rightarrow \mathbb{R}$$

$$h \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = g \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} / 2$$

Ruby

$$h = g / 2$$

MultiArray.dfloat

Fixnum

```
MultiArray.dfloat( 320, 240 ):
[[ 245.0, 244.0, 197.0, ... ],
 [ 245.0, 247.0, 197.0, ... ],
 [ 247.0, 248.0, 187.0, ... ]
 ...
```

MultiArray.respond_to?("binary_div_lint_dfloat")

no

h = g.collect { |x| x / 2 }

yes

[3.141]

Array.pack("D")

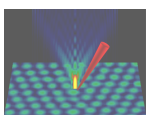
String.unpack("D")

C++

```
for ( int i=0; i<n; i++ )
    *r++ = *p++ / q;
```

```
MultiArray.binary_div_byte_byte
MultiArray.binary_div_byte_bytergb
MultiArray.binary_div_byte_dcomplex
MultiArray.binary_div_byte_dfloat
MultiArray.binary_div_byte_dfloatrgb
...
```

"\x54\xE3\xA5\x9B\xC4\x20\x09\x40"

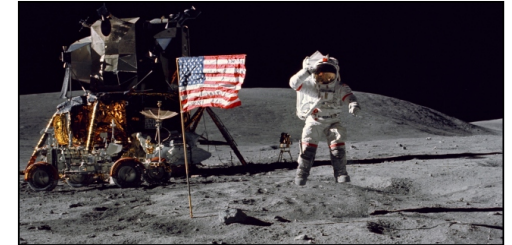


Generic operations

Linear shift-invariant filters

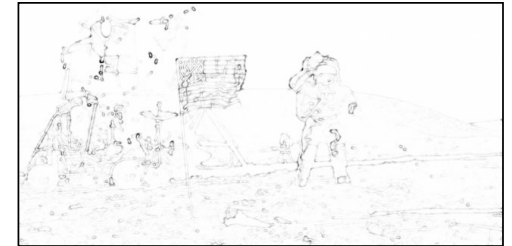
Common code

```
grad_sigma, cov_sigma = 1.0, 1.0
x, y = img.gauss_gradient_x( grad_sigma ), img.gaussgradient_y( grad_sigma )
a = ( x ** 2 ).gauss_blur( cov_sigma )
b = ( y ** 2 ).gauss_blur( cov_sigma )
c = ( x * y ).gauss_blur( cov_sigma )
tr = a + b
det = a * b - c * c
```



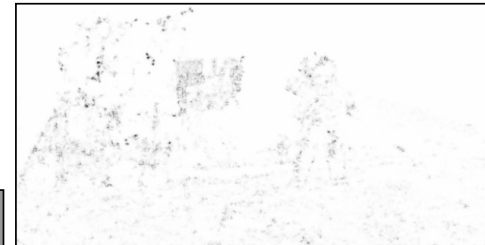
Yang et al.

```
noise = 1.0
g = ( ( a - b ) ** 2 + ( 2 * c ) ** 2 ) / ( a + b + noise ** 2 ) ** 2
result = ( g.normalise( 1.0..0.0 ) ** m * ( x ** 2 + y ** 2 ) )
```



Kanade-Lucas-Tomasi

```
dissqrt = ( tr ** 2 - det * 4 ).major( 0.0 ).sqrt
result = 0.5 * ( tr - dissqrt )
```



Harris-Stephens

```
k = 0.1
result = det - tr * tr * k
```



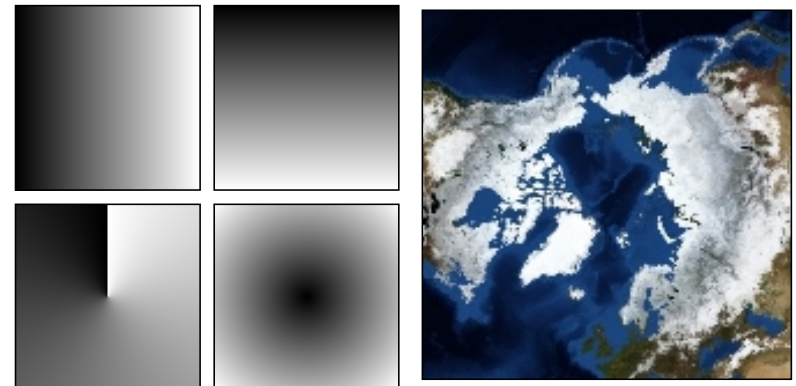
MultiArray.correlate_byte_byte
MultiArray.correlate_byte_bytergb
MultiArray.correlate_byte_dcomplex
MultiArray.correlate_byte_dfloat
MultiArray.correlate_byte_dfloatrb
...

Generic operations Warps

$$\begin{aligned}
 g &\in \{0, 1, \dots, w - 1\} \times \{0, 1, \dots, h - 1\} \rightarrow \mathbb{R}^3 \\
 h &\in \{0, 1, \dots, w' - 1\} \times \{0, 1, \dots, h' - 1\} \rightarrow \mathbb{R}^3 \\
 W &\in \{0, 1, \dots, w' - 1\} \times \{0, 1, \dots, h' - 1\} \rightarrow \mathbb{Z}^2
 \end{aligned}
 \quad
 h\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{cases} g(W\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right)) & \text{if } W\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) \in \{0, 1, \dots, w - 1\} \times \{0, 1, \dots, h - 1\} \\ 0 & \text{otherwise} \end{cases}$$

```

class MultiArray
def MultiArray.ramp1( *shape )
  retval = MultiArray.new( MultiArray::LINT, *shape )
  for x in 0...shape[0]
    retval[ x, 0...shape[1] ] = x
  end
  retval
end
# ...
end
img = MultiArray.load_rgb24( "test.jpg" )
w, h = *img.shape; c = 0.5 * h
x, y = MultiArray.ramp1( h, h ), MultiArray.ramp2( h, h )
warp = MultiArray.new( MultiArray::LINT, h, h, 2 )
warp[ 0...h, 0...h, 0 ], warp[ 0...h, 0...h, 1 ] =
  ( ( ( x - c ).atan2( y - c ) / Math::PI + 1 ) * w / 2 - 0.5 ),
  ( ( x - c ) ** 2 + ( y - c ) ** 2 ).sqrt
img.warp_clipped( warp ).display
  
```



Generic operations

Affine transforms

```

class MultiArray
  def MultiArray.ramp1( *shape )
    retval = MultiArray.new( MultiArray::LINT, *shape )
    for x in 0...shape[0]
      retval[ x, 0...shape[1] ] = x
    end
    retval
  end
  # ...
end
img = MultiArray.load_rgb24( "test.jpg" )
w, h = *img.shape
v = Vector[ MultiArray.ramp1( w, h ) - w / 2,
             MultiArray.ramp2( w, h ) - h / 2 ]
angle = 30.0 * Math::PI / 180.0
m = Matrix[ [ Math::cos( angle ), -Math::sin( angle ) ],
             [ Math::sin( angle ), Math::cos( angle ) ] ]
warp = MultiArray.new( MultiArray::LINT, w, h, 2 )
warp[ 0...w, 0...h, 0 ], warp[ 0...w, 0...h, 1 ] =
  ( m * v )[0] + w / 2, ( m * v )[1] + h / 2
img.warp_clipped( warp ).display
  
```

$$W_{\alpha} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$



given: template T , image I , previous pose \vec{p}

sought: pose-change $\Delta\vec{p}$

$$\operatorname{argmin}_{\Delta\vec{p}} \int_{\vec{x} \in T} \|T(\vec{x}) - I(W_{\vec{p}}^{-1}(W_{\Delta\vec{p}}^{-1}(\vec{x})))\|^2 d\vec{x} = (*)$$

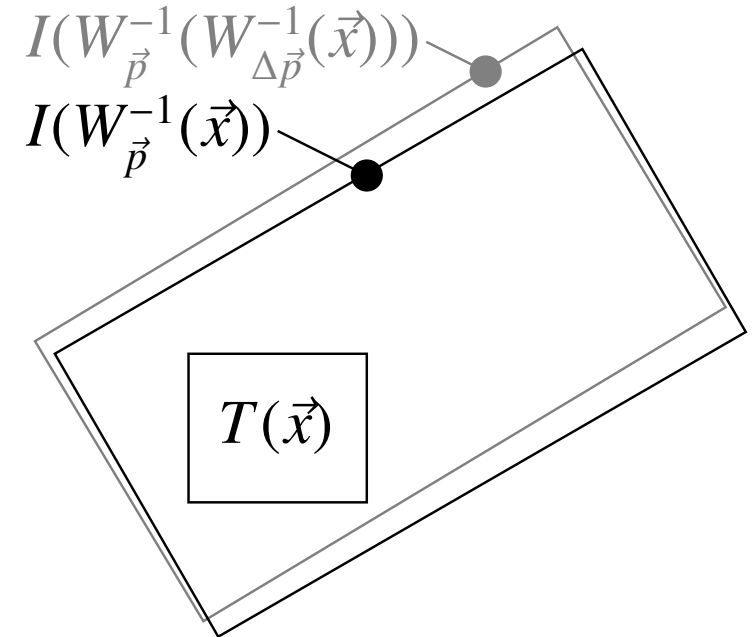
$$(1) T(\vec{x}) - I(W_{\vec{p}}^{-1}(W_{\Delta\vec{p}}^{-1}(\vec{x}))) = T(W_{\Delta\vec{p}}(\vec{x})) - I(W_{\vec{p}}^{-1}(\vec{x}))$$

$$(2) T(W_{\Delta\vec{p}}(\vec{x})) \approx T(\vec{x}) + \left(\frac{\delta T}{\delta \vec{x}}(\vec{x})\right)^T \cdot \left(\frac{\delta W_{\vec{p}}}{\delta \vec{p}}(\vec{x})\right) \cdot \Delta\vec{p}$$

$$(*) \stackrel{(1,2)}{\equiv} \operatorname{argmin}_{\vec{p}} (\|\mathcal{H} \vec{p} + \vec{b}\|^2) = (\mathcal{H}^T \mathcal{H})^{-1} \mathcal{H}^T \vec{b}$$

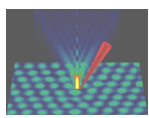
$$\text{where } \mathcal{H} = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots \\ h_{2,1} & h_{2,2} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \text{ and } \vec{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \end{pmatrix}$$

$$h_{i,j} = \left(\frac{\delta T}{\delta \vec{x}}(\vec{x}_i)\right)^T \cdot \left(\frac{\delta W_{\vec{p}}}{\delta p_j}(\vec{x}_i)\right), b_i = T(\vec{x}_i) - I(W_{\vec{p}}^{-1}(\vec{x}_i))$$



S. Baker and I. Matthew: "Lucas-Kanade 20 years on: a unifying framework"

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

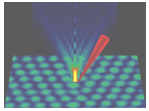


initialisation

```
p = Vector[ xshift, yshift, rotation ]  
w, h, sigma = tpl.shape[0], tpl.shape[1], 5.0  
x, y = xramp( w, h ), yramp( w, h )  
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

```
field = MultiArray.new( MultiArray::SFLOAT, 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_interpolate( field ) - 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
```

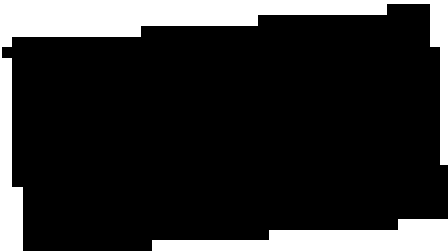



Application

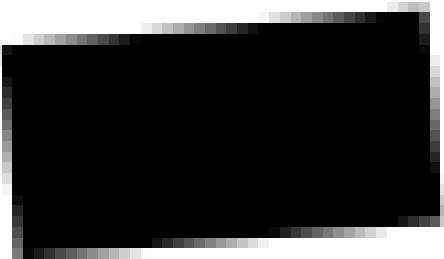
Lucas-Kanade implementation details

Interpolation

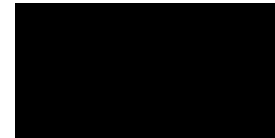
without interpolation



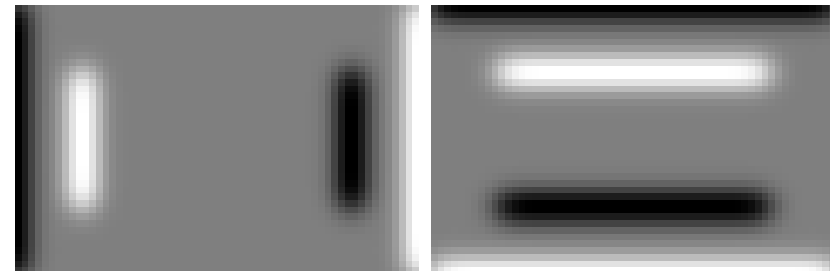
with interpolation



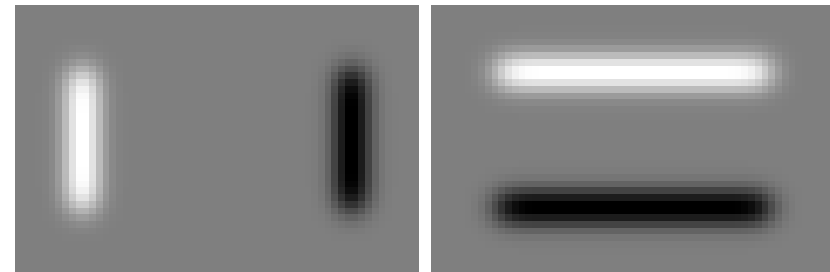
Gradient boundary



boundary effects



no boundary effects



conclusion

- concise implementation of Lucas-Kanade
- native blocks of code in C++, Ruby as glue-code
- development platform for general purpose machine vision

future work

- microscopy software
- wavelets, feature extraction
- feature-based object tracking & recognition

free software

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

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

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

