http://vision.eng.shu.ac.uk/jan/shef11.pdf

Computer vision with special reference to Ruby

Computer vision with special reference to Ruby

Jan Wedekind

Wed Apr 06 14:00:00 BST 2011

http://vision.eng.shu.ac.uk/jan/shef11.pdf

Ruby: Open Classes and Closures

```ruby
# open classes
class Fixnum
  def times_three
    3 * self
  end
end
4.times_three
# 12
```

```ruby
File.open('test.txt', 'r') do |f|
  3.times { puts f.readline }
end
```

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

http://vision.eng.shu.ac.uk/jan/shef11.pdf

Ruby: Ruby Extensions

```
// gcc -shared -fPIC -I/usr/lib/ruby/1.8/x86_64-linux
 //   -o myextension.so myextension.c
#include <ruby.h>
#include <math.h>

VALUE wrap_logx( VALUE self, VALUE x )
{
  return rb_float_new( log( NUM2DBL( self ) ) / log( NUM2DBL( x ) ) );
}

void Init_myextension(void) {
  VALUE numeric = rb_const_get( rb_cObject, rb_intern( "Numeric" ) );
  rb_define_method( numeric, "logx", RUBY_METHOD_FUNC( wrap_logx ), 1 );
}
```

```
#!/usr/bin/env ruby
require 'myextension'
e = 1024.logx( 2 )
puts "2 ** #{e} = 1024"
```

Example: Histogram-Based Classification

http://vision.eng.shu.ac.uk/jan/shef11.pdf

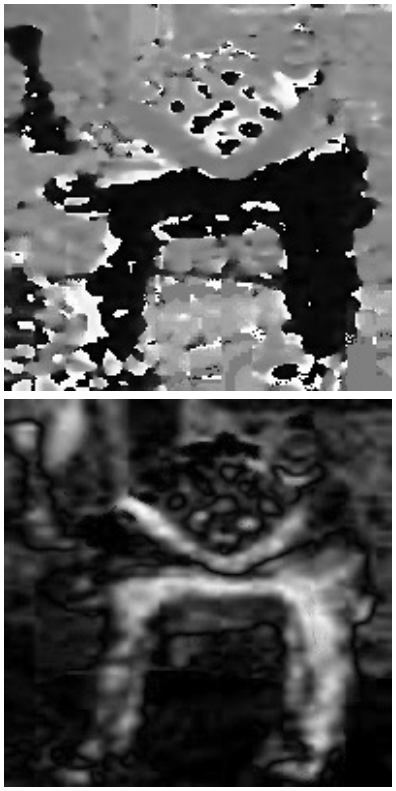
Hue-Saturation

$$\alpha = \tfrac{1}{2}(2R - G - B)$$

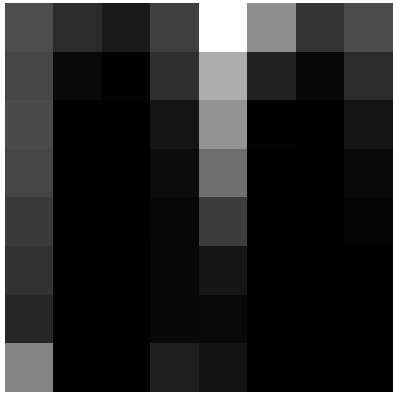
$$\beta = \tfrac{sqrt3}{2}(G - B)$$

$$H \approx \text{atan2}(\beta, \alpha)$$

$$C \approx \sqrt{\alpha^2 + \beta^2}$$



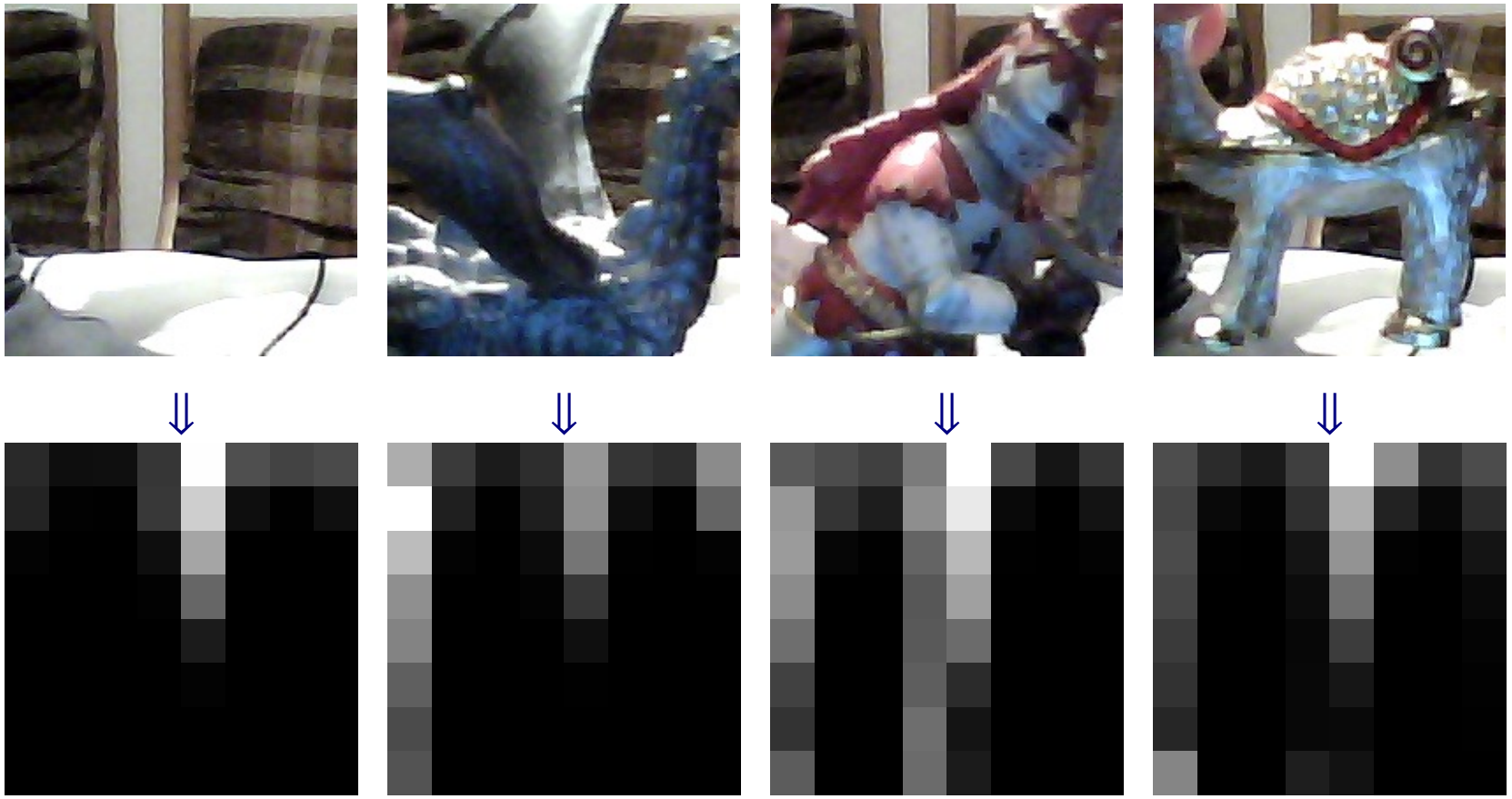
histogram



http://en.wikipedia.org/wiki/HSL_and_HSV

Similarity Measure

$$similarity(a,b) := \sum_{i,j} \left| \frac{a_{i,j} - b_{i,j}}{a_{i,j} + b_{i,j}} \right|$$



http://vision.eng.shu.ac.uk/jan/shef11.pdf

Source Code

```
MAXC = 117.0
HBINS, CBINS = 8, 8
THRESHOLD, N = 32, 5
BOX = [220 ... 420, 140 ... 340]
class Node
  def hsv_hist(hbins = HBINS, cbins = CBINS)
    alpha = 2 * r.to_sint - g - b
    beta = Math.sqrt(3) / 2 * ( g.to_sint - b )
    h = ( (Math.atan2(beta, alpha) + PI) * (HBINS / PI2) ).to_int.clip 0 .. HBINS - 1
    c = ( Math.sqrt(alpha ** 2 + beta ** 2) * (CBINS / MAXC) ).to_int.clip 0 .. CBINS - 1
    [h, c].histogram(hbins, cbins).to_int
  end
  def cmp(other)
    (self - other) / (self + other).major(1.0)
  end
end
input = V4L2Input.new
mask = MultiArray.bool(640, 480).fill!
mask[*BOX] = true
labels = [ 'reference' , 'dragon' , 'knight' , 'camel' ]
hists = labels.collect { |object| MultiArray.load_ubytergb( "#{object}.jpg" ).hsv_hist }
history = [ 'reference' ] * N
X11Display.show do
  img = input.read_ubytergb
  img_hist = img[*BOX].hsv_hist
  similarities = hists.collect { |hist| img_hist.cmp(hist).abs.sum }
  label = labels[similarities.index(similarities.min)]
  history = [label] + history[0 ... N - 1]
  if history == [label] * N
    system "echo '#{label}' | festival --tts" if label != 'reference'
  end
  history != [ 'reference' ] * N ? mask.conditional(img, img >> 1) : img
end
```