

# Integral Image

OpenCV顔抽出のツボ

[@maaash](#)

# 経緯

2008年流行ってたこと

OpenCVを使った顔抽出

イタリア支社で、それをFlashに移植してみようと思いい立ちました

<http://www.libspark.org/wiki/mash/Marilena>

サンプル

[http://maaash.jp/lab/Marilena\\_sample/](http://maaash.jp/lab/Marilena_sample/)

# 顔抽出

顔抽出の肝心部分

<http://www.libspark.org/browser/as3/Marilena/trunk/src/jp/maaash/ObjectDetection/ObjectDetector.as>

[line:66](http://www.libspark.org/browser/as3/Marilena/trunk/src/jp/maaash/ObjectDetection/ObjectDetector.as)

疑似コード

```
for ( factor=1; factor<limit; factor *= 1.2 ) {
    scaled_w = base_w * factor
    scaled_h = base_h * factor
    for ( x=0; x<image_w; x+= stepx ) {
        for ( y=0; y<image_y; y+= stepy ) {
            part_image = (x, y, scaled_w, scaled_h);
            if ( 予め与えられたパターンとpart_imageがマッチするか ) {
                push( @faces, part_image );
            }
        }
    }
}
```

# 顔抽出

すごい力技

でも、30fpsとか出てすごい

# マッチングさせるところ

マッチングさせる  
対象  
haarcascade.xml

顔の特徴を  
羅列してある  
これが人の顔である

```
<opencv_storage>
<haarcascade_frontalface_alt type_id="opencv-haar-classifier">
  <size>20 20</size>
  <stages>
    <_>
      <!-- stage 0 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>3 7 14 4 -1.</_>
                <_>3 9 14 2 2.</_></rects>
              <tilted>0</tilted></feature>
              <threshold>4.0141958743333817e-003</threshold>
              <left_val>0.0337941907346249</left_val>
              <right_val>0.8378106951713562</right_val></_></_>
          <_>
            <!-- tree 1 -->
            <_>
              <!-- root node -->
              <feature>
                <rects>
                  <_>1 2 18 4 -1.</_>
                  <_>7 2 6 4 3.</_></rects>
                <tilted>0</tilted></feature>
                <threshold>0.0151513395830989</threshold>
                <left_val>0.1514132022857666</left_val>
                <right_val>0.7488812208175659</right_val></_></_>
            <_>
              <!-- tree 2 -->
              <_>
                <!-- root node -->
                <feature>
                  <rects>
                    <_>1 7 15 9 -1.</_>
                    <_>1 10 15 3 3.</_></rects>
                  <tilted>0</tilted></feature>
                  <threshold>4.2109931819140911e-003</threshold>
                  <left_val>0.0900492817163467</left_val>
                  <right_val>0.6374819874763489</right_val></_></_></trees>
            <stage_threshold>0.8226894140243530</stage_threshold>
            <parent>-1</parent>
            <next>-1</next></_>
          </_>
        </_>
      </_>
    </_>
  </_>
</opencv_storage>
```

# マッチングさせるところ

顔の特徴(feature)とは

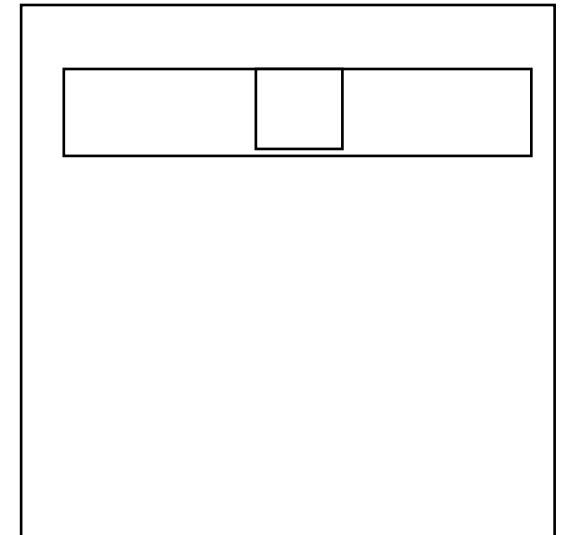
- ・正方形の中のある矩形領域 (x,y,w,h)
- ・その領域の重さ(weight)
- ・しきい値(th)
- ・left\_val, right\_val

疑似コード

```
val = 0;
while ( feature = features.next ) {
  val +=
    (featureの矩形領域範囲内の明度の合計 x weight > th)
      ? right_val
      : left_val
}
if ( val > 顔であるというしきい値 ) { return 1; }
```

```
<_>
<!-- tree 1 -->
<_>
<!-- root node -->
<feature>
<rects>
  <_>1 2 18 4 -1.</_>
  <_>7 2 6 4 3.</_></rects>
<tilted>0</tilted></feature>
<threshold>0.0151513395830989</threshold>
<left_val>0.1514132022857666</left_val>
<right_val>0.7488812208175659</right_val></_></_>
<_>
```

width:20, height: 20



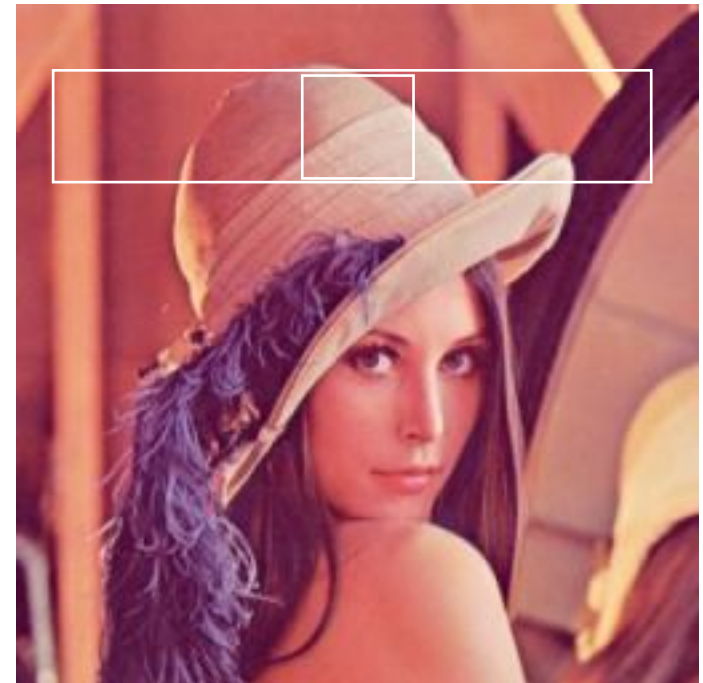
# 1つの特徴とマッチングさせる

```
<_>
<!-- tree 1 -->
<_>
  <!-- root node -->
  <feature>
    <rects>
      <_>1 2 18 4 -1.</_>
      <_>7 2 6 4 3.</_></rects>
    <tilted>0</tilted></feature>
  <threshold>0.0151513395830989</threshold>
  <left_val>0.1514132022857666</left_val>
  <right_val>0.7488812208175659</right_val></_></_>
<_>
```

width:20, height: 20

(1,2,18,4)の範囲内の画素の明るさを合計したもの $\times -1 +$   
(7,2,6,4)の範囲内の画素の明るさを合計したもの  $\times 3 \Rightarrow$  sum

```
val = 0;
if ( sum > 0.0151.. ) {
  val += 0.74888...;
}
else {
  val += 0.151413...;
}
```



# Integral Image

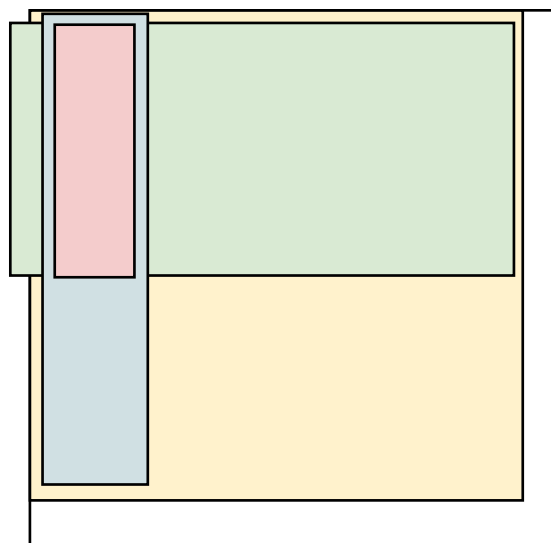
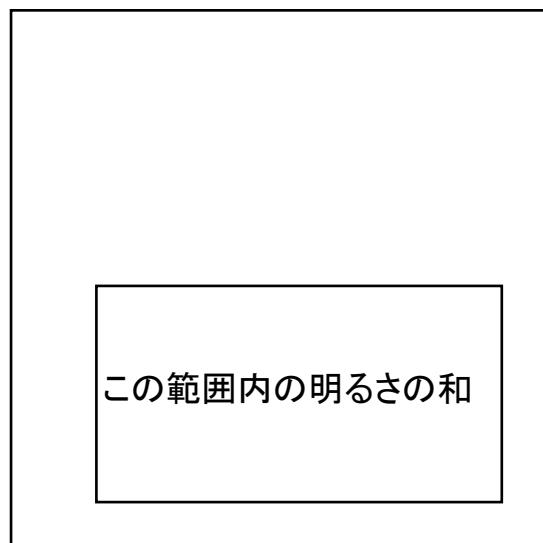
ある矩形範囲内の明るさ(なんでもいい)の和  
を高速に合計するための計算方法

<http://www.volksbot.de/surmann/papers/icra2005/node12.html>

[http://en.wikipedia.org/wiki/Integral\\_image](http://en.wikipedia.org/wiki/Integral_image)

$I(x, y) = I(x, y - 1) + I(x - 1, y) + N(x, y) - I(x - 1, y - 1)$  with  $I(-1, y) = I(x, -1) = I(-1, -1) = 0$ ,

$$F(x, y, h, w) = I(x, y) + I(x + w, y + h) - I(x, y + h) - I(x + w, y).$$



この範囲内の明るさの和  
= 黄 + 赤 - 青 - 緑

# Integral Image

あらかじめIntegralImage:  $I(x,y)$  を計算しておけば、  
(これも  $0 < x < w, 0 < y < h$  の一度の走査と加減算で済む)  
矩形範囲内の明るさの合計:  $F(x,y,h,w)$  はI配列への4回の参照でok

$$F(x,y,h,w) = I(x,y) + I(x+w,y+h) - \\ I(x,y+h) - I(x+w,y).$$

# Integral Image

応用してね