Imagem integral (ou integral da imagem)

Imagem f.

8	3	9	7
1	8	5	2
8	7	3	6
5	2	9	3

`s: Integral da imagem f (imagem integral)`

8	11	20	27
9	20	34	43
17	35	52	67
22	42	68	86

s(u,v) = f(u,v)+s(u-1,v)+s(u,v-1)-s(u-1,v-1)
with s(u,v) = 0 when either u, v < 0.
Para calcular a soma de f dentro do retângulo (li,ci) até (lf,cf):
soma = s(lf,cf) - s(li-1,cf) - s(lf,ci-1) + s(li-1,ci-1)
Nota: backg=0
Exemplo: (li,ci)=(1,1) (lf,cf)=(2,2). Soma=52-20-17+8=23 (8+5+7+3=23)

Imagem f^2.

64	9	81	49
1	64	25	4
64	49	9	36
25	4	81	9

s2: Integral do quadrado da imagem f

64	73	154	203
65	138	244	297
129	251	366	455
154	280	476	574

s2(u,v) = f2(u,v)+s2(u-1,v)+s2(u,v-1)-s2(u-1,v-1)
with s2(u,v) = 0 when either u, v < 0.
Para calcular a soma de f ao quadrado dentro do retângulo (li,ci) até (lf,cf):
soma2 = s2(lf,cf) - s2(li-1,cf) - s2(lf,ci-1) + s2(li-1,ci-1)
Nota: backg=0
Exemplo: (li,ci)=(1,2) (lf,cf)=(2,2). Soma2=366-154-251+73=34 (25+9=34)

Em OpenCV:

```
//integral.cpp pos2015
#include <cekeikon.h>
int main()
{ Mat_<GRY> g=(Mat_<GRY>(4,4) << 8,3,9,7,
                                  1,8,5,2,
                                  8,7,3,6,
                                  5,2,9,3);
  Mat_<int> s,ti; Mat_<double> sq;
  //integral(g, s);
  //integral(g, s, sq);
  integral(g, s, sq, ti);
  cout << s << endl;
  cout << sq << endl;
  cout << ti << endl;
}
```

```
g=
[8, 3, 9, 7;
 1, 8, 5, 2;
 8, 7, 3, 6;
 5, 2, 9, 3]

s=
[0,  0,  0,  0,  0;
 0,  8, 11, 20, 27;
 0,  9, 20, 34, 43;
 0, 17, 35, 52, 67;
 0, 22, 42, 68, 86]
```

* Para calcular a soma de f dentro do retângulo (li,ci) inclusive até (lf,cf) exclusive:
soma = s(lf,cf) - s(li,cf) - s(lf,ci) + s(li,ci)
Exemplo: (li,ci)=(1,1) (lf,cf)=(3,3).
Soma = s(3,3) - s(1,3) - s(3,1) + s(1,1) = 52-20-17+8 = 23 (8+5+7+3=23)

```
sq=
[0,   0,   0,   0,   0;
 0,  64,  73, 154, 203;
 0,  65, 138, 244, 297;
 0, 129, 251, 366, 455;
 0, 154, 280, 476, 574]
```

* Para calcular a soma de g ao quadrado dentro do retângulo (li,ci) inclusive até (lf,cf) exclusive:
soma2 = s2(lf,cf) - s2(li,cf) - s2(lf,ci) + s2(li,ci)
Exemplo: (li,ci)=(1,2) (lf,cf)=(3,3).
Soma2=s2(3,3) - s2(1,3) - s2(3,2) + s2(1,2) = 366-154-251+73=34 (25+9=34)

* Para calcular o desvio-padrão σ dentro do retângulo (li,ci) inclusive até (lf,cf) exclusive:
Exemplo: (li,ci)=(0,0) (lf,cf)=(2,2).
Soma = 20 Média μ=5
Soma_quadrática = 138 Média_quadrática ν=5.87

$$\sigma = \sqrt{\nu^2 - \mu^2} = 3.08$$

raiz[$[(8-5)^2 + (3-5)^2 + (1-5)^2 + (8-5)^2]$ / 4] = raiz[(9+4+16+9)/4] = 3.08

* Para calcular o desvio-padrão σ dentro do retângulo (li,ci) inclusive até (lf,cf) exclusive:
Exemplo: (li,ci)=(1,1) (lf,cf)=(3,3).
Soma = 23 Média μ=5.75
Soma_quadrática = 366-129-154+64 = 147 Média_quadrática ν=6.062

$$\sigma = \sqrt{\nu^2 - \mu^2} = 1.92$$

raiz[$[(8-5.75)^2 + (5-5.75)^2 + (7-5.75)^2 + (3-5.75)^2] / 4$] = 1.92

```
g=
[8, 3, 9, 7;
 1, 8, 5, 2;
 8, 7, 3, 6;
 5, 2, 9, 3]

t=
[ 0,  0,  0,  0,  0;
  0,  8,  3,  9,  7;
  8, 12, 28, 24, 18;
 12, 37, 48, 45, 32;
 37, 61, 63, 68, 54]
```

* Para calcular a soma de g dentro do retângulo (l,c) dimensão (h,w) – h para esquerda, w para direita:

tilt = t(l+w+h+2,c+w-h+1) + t(l,c+1) - t(l+h+1,c-h) + t(l+w+1,c+w+2)

Exemplo: (l,c)=(0,1) (h,w)=(1,1).

tilt=3+1+8+5+8+7+3+2=37

tilt= t(4,2) + t(0,2) - t(2,0) - t(2,4) = 63 + 0 - 8 -18 = 37

Nota: Pode precisar de mais uma coluna à direita.

A imagem t é calculada em dois passos.

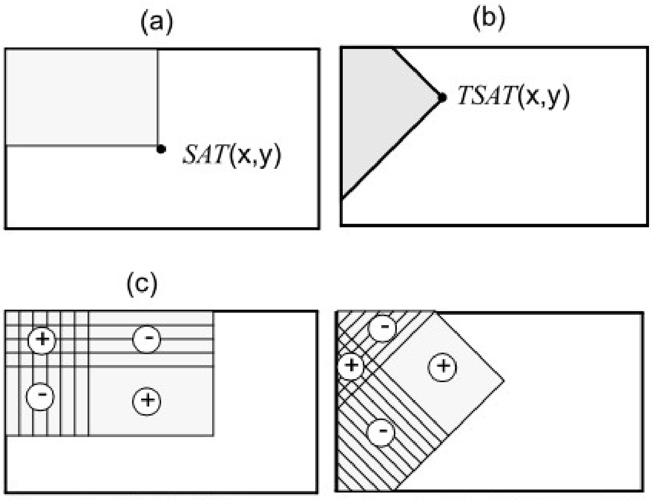


Figure 3. (a) Upright *Summed Area Table* (*SAT*) and (b) *Rotated Summed Area Table* (*RSAT*); calculation scheme of the pixel sum of upright (c) and rotated (d) rectangles.

For 45° rotated rectangles the auxiliary image is defined as the *Rotated Summed Area Table* $RSAT(x,y)$. It gives the sum of the pixels of the rectangle rotated by 45° with the right most corner at (x,y) and extending till the boundaries of the image (see Figure 3b):

$$RSAT(x,y) = \sum_{x' \le x,\, x' \le x-|y-y'|} I(x',y') \,.$$

It can be calculated with two passes over all pixels. The first pass from left to right and top to bottom determines

$$RSAT(x,y) = RSAT(x-1,y-1) + RSAT(x-1,y) + I(x,y) - RSAT(x-2,y-1)$$

with

$$RSAT(-1,y) = RSAT(-2,y) = RSAT(x,-1) = 0 \,,$$

whereas the second pass from the right to left and bottom to top calculates

$$RSAT(x,y) = RSAT(x,y) + RSAT(x-1,y+1) - RSAT(x-2,y)$$

From this the pixel sum of any rotated rectangle $r = (x,y,w,h,45°)$ can be determined by four table lookups (see also Figure 3(d) and Figure 4):

$$\begin{aligned} RecSum(r) = {} & RSAT(x+w,y+w) + RSAT(x-h,y+h) \\ & - RSAT(x,y) - RSAT(x+w-h,y+w+h) \,. \end{aligned}$$

Outro exemplo maior para calcular integral rotacionado:

```
f=
[        8,   3,   9,   7,   3,   7,   5,   4;
         1,   8,   5,   2,   2,   5,   3,   9;
         8,   7,   3,   6,   7,   5,   1,   3;
         5,   2,   9,   3,   4,   2,   3,   6;
         2,   3,   4,   8,   9,   5,   7,   3;
         7,   3,   1,   2,   3,   2,   4,   5;
         3,   4,   8,   6,   9,   7,   2,   3;
         7,   5,   9,   7,   5,   3,   5,   4]

t=
[   0,   0,   0,   0,   0,   0,   0,   0,   0;
    0,   8,   3,   9,   7,   3,   7,   5,   4;
    8,  12,  28,  24,  21,  19,  20,  19,  18;
   12,  37,  48,  48,  44,  47,  41,  37,  31;
   37,  61,  66,  80,  83,  77,  71,  57,  46;
   61,  73,  98, 114, 124, 120, 100,  90,  66;
   73, 107, 127, 147, 161, 159, 146, 120,  98;
  107, 137, 163, 183, 190, 199, 188, 160, 128;
  137, 173, 202, 223, 234, 233, 223, 203, 167]
```

Somatória no losango = 79

Soma = 190 - 57 - 61 + 7 =79

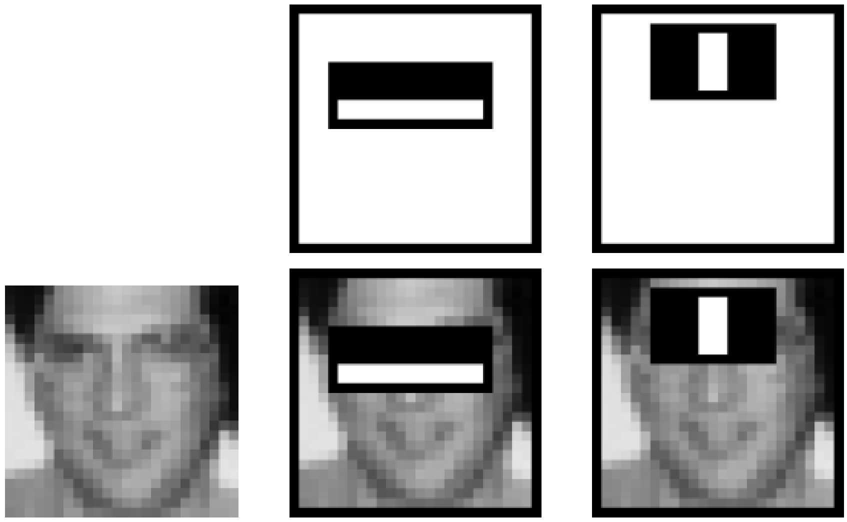
Imagem integral é usado (por exemplo) em:

1) Calcular rapidamente NCC.
2) Na detecção de rostos (Viola e Jones).

Bibliografia para fast NCC:
[Lewis1995] J.P. Lewis, “Fast normalized cross-correlation,” *Vision Interface*, pp. 120-123, 1995, url = "citeseer.ist.psu.edu/lewis95fast.html"

Bibliografia para detecção de rostos usando imagem integral:
[Viola2004] P. Viola, M. J. Jones, “Robust Real-Time Face Detection,” *International Journal of Computer Vision* 57(2), 137–154, 2004.
[Viola2001] P. Viola, M. J. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” Conference on Computer Vision and Pattern Recognition, 2001.
[Lienhart] R. Lienhart, J. Maydt, “An Extended Set of Haar-like Features for Rapid Object Detection,” Int. Conf. on Image Processing, 2002.

Programa exemplo do OpenCV para detecção de rostos:
C:\Opencv2410\build\x86\MinGW-SE\samples\c>c-example-facedetect.exe
Precisa rodar do diretório:
C:\Opencv2410\sources\data\haarcascades

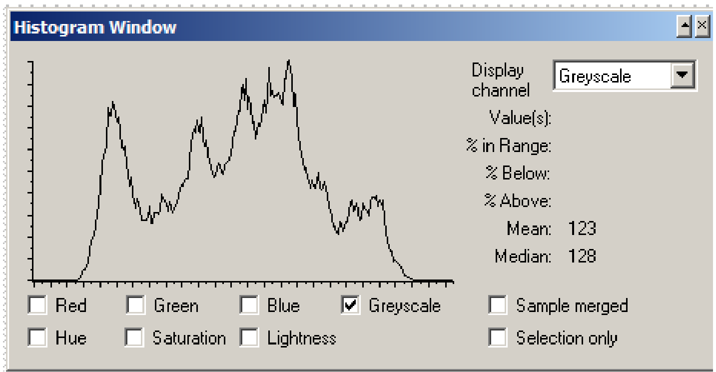


Histograma de níveis de cinza

Calcular histograma de uma imagem:

```
//histog.cpp - pos2012
#include <cekeikon.h>
int main()
{ Mat_<GRY> a; le(a,"lennag.tga");
  vector<int> histog(8,0);
  // vetor de 8 posicoes preenchida com zero
  for (int l=0; l<a.rows; l++)
    for (int c=0; c<a.cols; c++) {
      int i=a(l,c)/32; // i pertence [0,8[
      histog[i]++;
    }
  cout << histog << endl;
}

saída:
[415, 42487, 31329, 54730, 73740, 35218, 23815, 410]
```

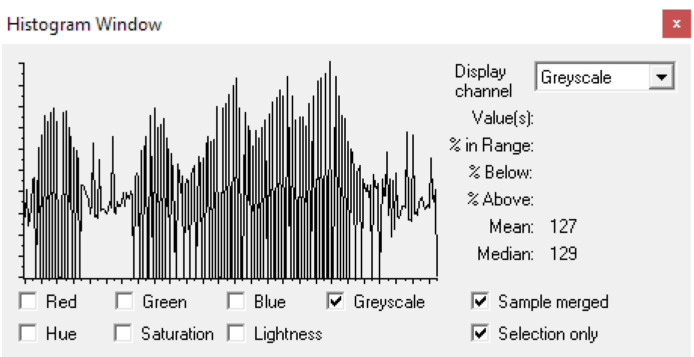
lennag



histograma de níveis de cinza

Equalizar histograma

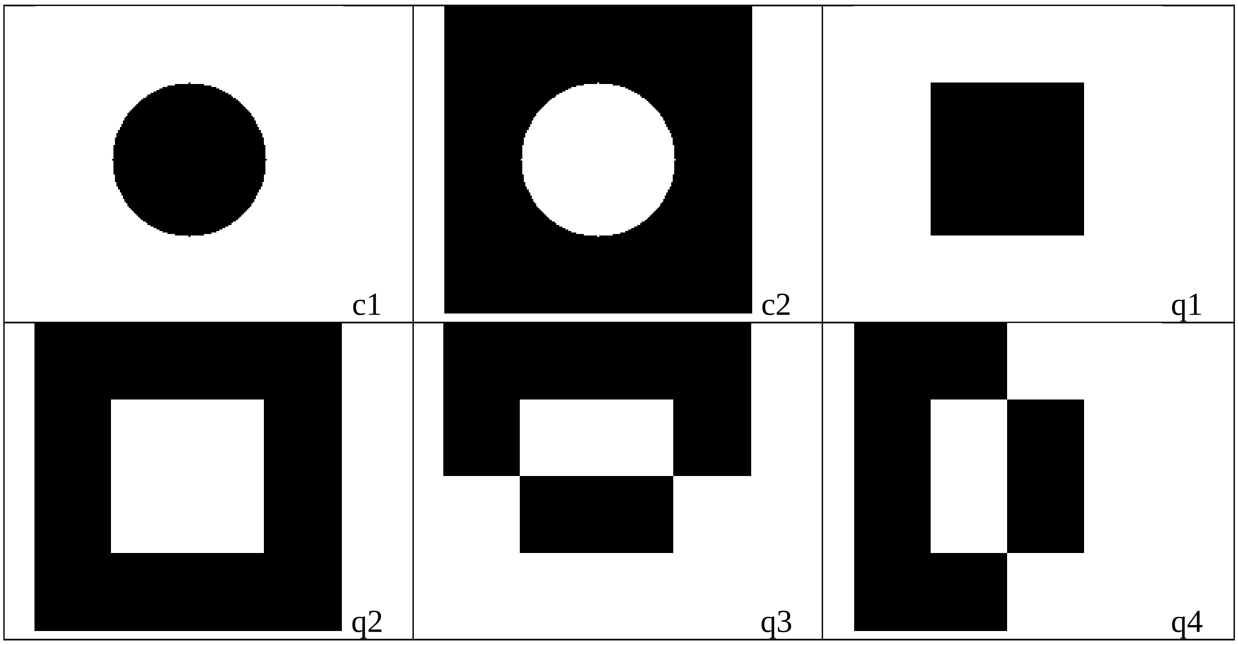
equalização de histograma



histograma de níveis de cinza

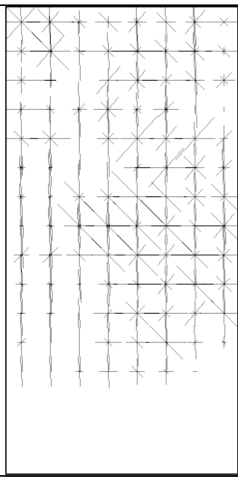
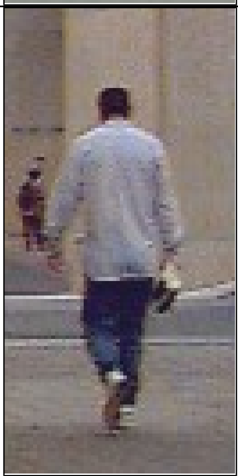
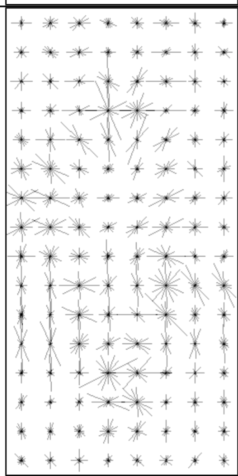
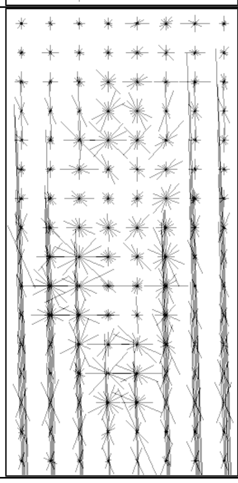
Calcular histograma de gradiente orientado (HOG) de uma imagem:

```
//hog.cpp pos2014
#include <cekeikon.h>

int main(int argc, char** argv)
{ if (argc!=2) erro("Erro: hog nome.pgm");
  Mat_<FLT> a; le(a,argv[1]);
  GaussianBlur( a, a, Size(17,17),0,0); // Isto e' necessario
  //mostra(a);

  Mat_<FLT> gx=filtro2d(a, (Mat_<FLT>(1,3) << -1,0,1) );
  Mat_<FLT> gy=filtro2d(a, (Mat_<FLT>(3,1) << -1,0,1) );

  int nbins=8;
  vector<double> uhog(nbins, 0.0); // HOG sem sinal
  vector<double> shog(nbins, 0.0); // HOG com sinal
  for (int l=1; l<gx.rows-1; l++)
    for (int c=1; c<gx.cols-1; c++) {
      double graus=fastAtan2(gy(l,c),gx(l,c));
      double modulo=sqrt(elev2(gx(l,c))+elev2(gy(l,c)));
      int si = int(graus/(360.0/nbins));
      if (graus>=180) graus -= 180; // graus entre [0 e 180[
      int ui = int(graus/(180.0/nbins));
      uhog[ui] += modulo;
      shog[si] += modulo;
    }
  cout << uhog << endl;
  cout << shog << endl;
}
```



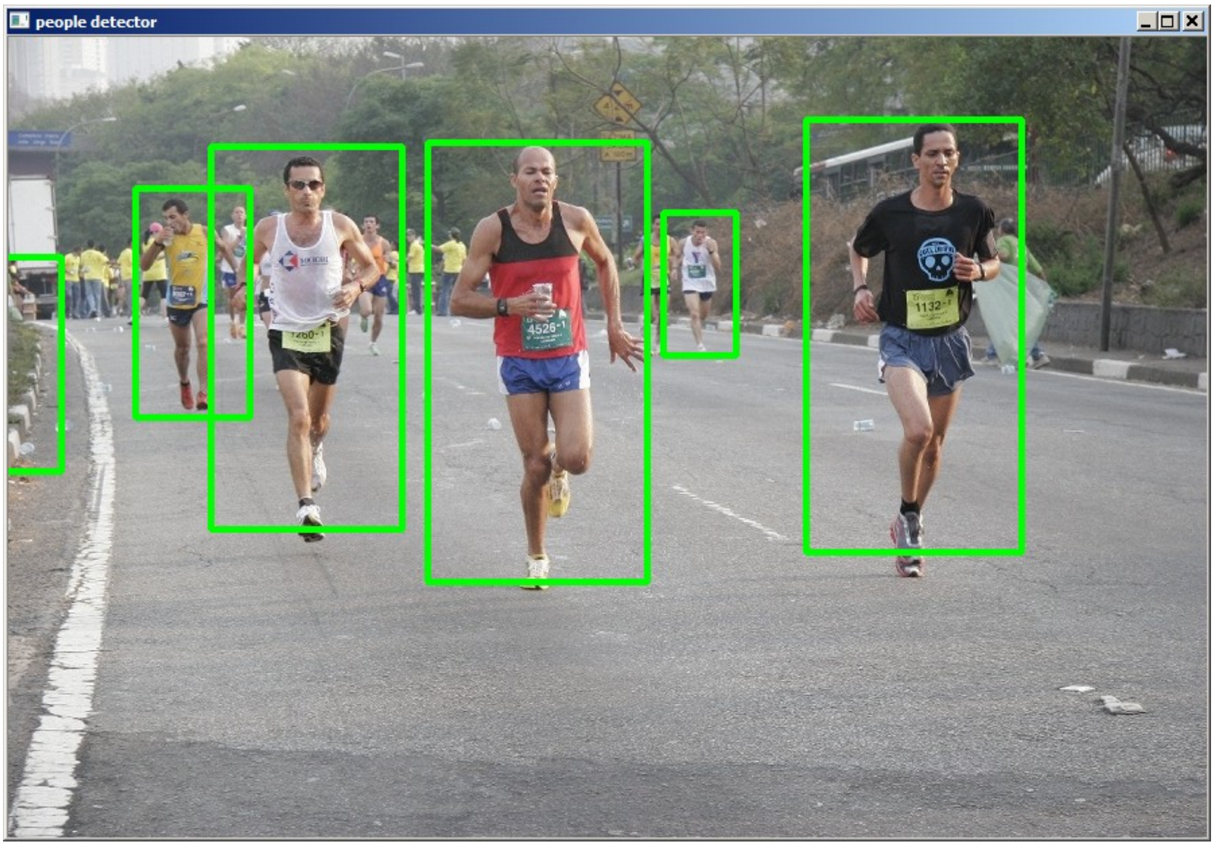

```
Sem sinal (vai de 0 graus até 180 graus).
(graus)  0      22.5      45     67.5      90     112.5     135      157
C1 [73.2208, 90.2259, 81.5353, 68.9685, 73.2157, 81.5353, 90.2259, 68.9736]
C2 [73.2208, 90.2259, 81.5353, 68.9685, 73.2157, 81.5353, 90.2259, 68.9736]
Q1 [367.964, 6.10659, 3.27816, 15.7526, 367.964, 3.27816, 6.10659, 15.7526]
Q2 [367.964, 4.80807, 4.57668, 15.7526, 367.964, 4.57668, 4.80807, 15.7526]
Q3 [332.341, 9.97787, 5.93412, 28.1293, 730.341, 5.93412, 9.97787, 28.1293]
Q4 [730.341, 9.97787, 5.93412, 28.1293, 332.341, 5.93412, 9.97787, 28.1293]
```

Calcular histograma de gradiente orientado (HOG) de uma imagem por pedaços:
`pos2012`

Identificar se uma imagem é de pedestre ou de automóvel
Solução em: c:\haepi\algpi\mle\pedestre
Veja a apostila de aprendizagem

c:\opencv2410\build\x86\MinGW-SE\samples\cpp>cpp-example-peopledetect.exe \lixo_mg_0113.jpg



Acelerar o cálculo de HOG usando imagem integral:

O cálculo de HOG para qualquer janela retangular W de uma imagem grande Q pode ser acelerado calculando primeiro o integral do HOG de Q. Para isso:

1) Considere, por exemplo, 4 bins. Suponha que queremos calcular HOG sem sinal. Vamos criar 4 imagens integrais I_0, I_1, I_2, I_3 para as 4 bins.
2) Calcule o gradiente ∇Q para todos os pixels de Q.
3) Calcule as imagens F_0, F_1, F_2, F_3. $F_i(p)=|\nabla Q(p)|$ se o ângulo do gradiente em p cair no bin i. $F_i(p)=0$ caso contrário.
4) Calcule as imagens integrais I_0, I_1, I_2, e I_3 das imagens F_0, F_1, F_2, F_3.
5) Agora, o integral do HOG de Q está armazenado nas quatro imagens I_0, I_1, I_2, I_3.

Para calcular HOG de uma janela retangular W de Q, basta calcular as somas dentro de W das quatro imagens integrais I_0, I_1, I_2, e I_3.

HOG de imagens coloridas:

Perde-se informação importante se converter uma imagem colorida para níveis de cinza antes de calcular HOG. Pois, duas cores diferentes podem ser mapeadas no mesmo nível de cinza.

Uma forma de evitar esta perda de informação (sem multiplicar o número de features por 3) é utilizar:

1) Seja uma imagem colorida Q com as três bandas Q_R, Q_G, Q_B.
2) Calcule os gradientes das três bandas ∇Q_R, ∇Q_G, ∇Q_B.
3) Para cada pixel p, calcule a banda $i \in \{R, G, B\}$ que tem o maior módulo de gradiente em p. Isto é, $|\nabla Q_i(p)| = \max\{ |\nabla Q_R(p)|, |\nabla Q_G(p)|, |\nabla Q_B(p)| \}$.
4) Considere como gradiente em p o gradiente da banda i. Isto é, $\nabla Q(p) = \nabla Q_i(p)$.

Em outras palavras, estamos considerando como gradiente de Q no pixel p o gradiente da banda com o maior módulo em p.