Introdução a Processamento de Imagens e Visão Computacional usando OpenCV

Tutorial 2

SIBGRAPI 2008

XXI BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING

CAMPO GRANDE/MS - BRAZIL

October 12-15, 2008

Introdução à Visão Computacional usando OpenCV

Maurício Marengoni
Denise Stringhini

Universidade Presbiteriana Mackenzie
Faculdade de Computação e Informática e
Pós Graduação em Engenharia Elétrica
{mmarengoni, dstring}@mackenzie.br

Resumo

Este tutorial apresenta conceitos introdutórios de processamento de imagens e de visão computacional.

Estes conceitos são introduzidos utilizando a biblioteca OpenCV, que é distribuída gratuitamente e possui documentação farta na internet.

Introdução

- Processamento de imagens é um processo onde a entrada do sistema é uma imagem e a saída é um conjunto de valores numéricos, que podem ou não compor uma outra imagem.
- Visão computacional procura emular a visão humana, portanto também possui como entrada uma imagem, porém, a saída é uma interpretação da imagem como um todo, ou parcialmente.

Introdução

Placa:
BRK 8558
Veículo:
Pajero 1995

Veículo em imagem escura.

Após uma equalização de histograma, em nível de cinza, onde a placa do veículo pode ser lida.

Informação da placa e do veículo no retângulo.

Introdução

- Processos de visão computacional geralmente iniciam com o processamento de imagens.
- Processamento ocorre em três níveis:
  - baixo-nível: operações primitivas (redução de ruído ou melhoria no contraste de uma imagem)
  - nível-médio: operações do tipo segmentação ou classificação
  - alto-nível: tarefas de cognição normalmente associadas com a visão humana

OpenCV

- OpenCV (*Open Source Computer Vision*) é uma biblioteca, de código aberto, desenvolvida inicialmente pela Intel.
- Implementa ferramentas de interpretação de imagens, indo desde operações simples como um filtro de ruído, até operações complexas, tais como a análise de movimentos, reconhecimento de padrões e reconstrução em 3D.

OpenCV

- Está dividida em cinco grupos de funções:
  - Processamento de imagens;
  - Análise estrutural;
  - Análise de movimento e *rastreamento* de objetos;
  - Reconhecimento de padrões;
  - Calibragem de câmera e reconstrução 3D.

As principais serão apresentadas juntamente com os conceitos de processamento de imagens e visão computacional que devem ser empregados em seu uso.

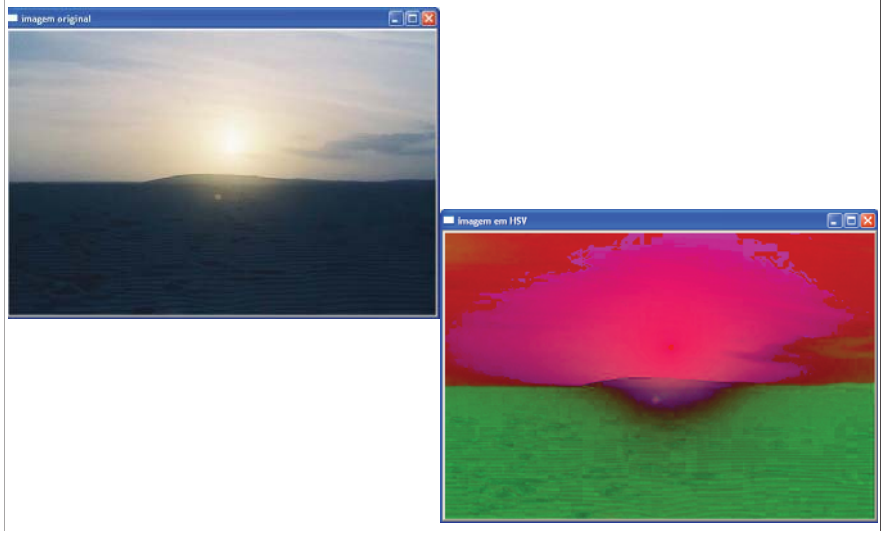
OpenCV - Exemplo


```
// conversaoCores.cpp - faz a conversao de
//cores de RGB para HSV
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
int main()
{
  char name[] = "imagens/exemplo1.jpg";
   // definicao de variaveis do tipo imagem
   IplImage* img = NULL;
   IplImage* res = NULL;
   // definicao de variavel que armazena uma
   // estrutura com o tamanho de uma imagem
   CvSize imgSize;
   // carrega a imagem exemplo1 na variavel img
   img = cvLoadImage( name, -1 );
   // extrai o tamanho da variavel carregada
   imgSize.width = img->width;
   imgSize.height = img->height;
   // cria uma nova imagem com o mesmo tamanho
   // da imagem carregada e do mesmo tipo
   res = cvCreateImage(imgSize, img->depth,img->nChannels );
```

```
// faz a conversao do sistema de cor RGB para HSV
   cvCvtColor( img, res, CV_RGB2HSV );
   // cria uma janela de visualizacao com o nome "imagem
      original"
   cvNamedWindow( "imagem original", 1 );
   // carrega a imagem original na janela criada
   cvShowImage( "imagem original", img );
   //repete o processo para a outra imagem
   cvNamedWindow( "imagem em HSV", 1 );
   cvShowImage( "imagem em HSV", res );
   // aguarda alguem pressionar uma tecla do teclado
   cvWaitKey(0);
   //fecha as janelas criadas
   cvDestroyWindow( "imagem original" );
   cvDestroyWindow( "imagem em HSV" );
   // libera memoria usada pelas imagens
   cvReleaseImage( &img );
   cvReleaseImage( &res );
   return(0);
}
```

OpenCV - Exemplo

8



Processamento de Imagens

9

- Muitas vezes as imagens de onde queremos extrair alguma informação precisam ser convertidas para um determinado formato ou tamanho.
- Precisam ainda ser filtradas para remover ruídos provenientes do processo de aquisição da imagem.

Ruídos

10

- Ruídos podem aparecer de diversas fontes.
  - Exemplos: tipo de sensor utilizado, iluminação do ambiente, condições climáticas no momento da aquisição da imagem, posição relativa entre o objeto de interesse e a câmera.
- Ruído não é apenas interferência no sinal de captura da imagem.
  - São também interferências que possam, atrapalhar a interpretação ou o reconhecimento de objetos na imagem.

Ruídos

11

Filtros

12

- Filtros são as ferramentas básicas para remover ruídos de imagens.
  - Neste caso, o ruído é aquele que aparece no processo de aquisição da imagem.
- Podem ser:
  - Espaciais: filtros que atuam diretamente na imagem.
  - De frequência: a imagem é transformada para o domínio de frequência (transformada de Fourier) e então é filtrada neste domínio. Em seguida a imagem filtrada é transformada de volta para o domínio de espaço.

Filtros

13

Do lado esquerdo uma imagem com ruído, e na direita a mesma imagem após filtragem.

Domínio de espaço

14

- O termo domínio espacial se refere à imagem.
  - Métodos no domínio espacial estão baseados na manipulação direta dos pixels da própria imagem.
- Os processos no domínio espacial são caracterizados pela seguinte expressão:

$$g(x,y) = T(f(x,y))$$

  - onde:

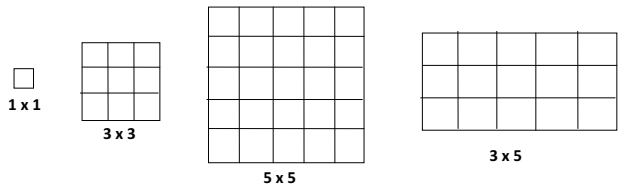
    f(x,y) é a imagem original,
    T(.) é uma transformação na imagem e
    g(x,y) é a imagem transformada.

Domínio de espaço

15

- T é uma operação definida sobre uma vizinhança de influência do pixel que está localizado na posição x, y.
- A idéia de vizinhança de influência considera os pixels ao redor da posição x, y.
- Esta vizinhança é definida por uma região quadrada (ou retangular) e de tamanho (lado) impar.

Domínio de espaço



Regiões de vizinhança (máscaras) de tamanhos diferentes, porém, todos com lado impar.

Função de transformação de intensidade

- O operador T é computado em uma vizinhança de tamanho 1x1.
  - É utilizada para alterar a intensidade da imagem.
  - Pode ser aplicada a toda a imagem ou a uma parte dela.
- Uma operação bastante útil é a binarização de uma imagem, que utiliza um certo valor de corte (k).

Função de transformação de intensidade

- Este tipo de transformação é definida pela seguinte expressão:

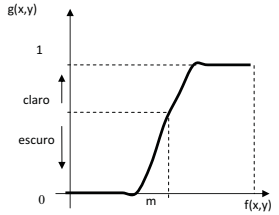
$$g(x,y) = \begin{cases} 1 & , \text{se } f(x,y) \geq k \\ 0 & , \text{caso contrario} \end{cases}$$

cvThreshold(ImgOrigem, ImgBinaria, 84, 255, CV_THRESH_BINARY);

Imagem binarizada, com o urso em destaque (maior bloco de pixels com valor 0) . A binarizacao foi obtida com o valor de k=84.

Função de realce de contraste

$$g(x,y) = \frac{1}{1 + (m/f(x,y))^E}$$



Onde, *m* é o valor médio da região que se deseja realçar o contraste, e *E* define a inclinação da curva, e conseqüentemente os valores de mapeamento para a imagem de saída.

Histogramas

- Os histogramas são determinados a partir de valores de intensidade dos pixels.
- Entre as principais aplicações dos histogramas estão
  - melhora da definição de uma imagem
  - compressão de imagens
  - segmentação de imagens
  - descrição de uma imagem

Histogramas

- O histograma de uma imagem I, cujos valores de intensidade estejam entre 0 e G, é definido pela expressão:

$$h(I_k) = n_k$$

Onde:
I_k é um valor de intensidade k, ($0 \le k \le G$) da imagem I
n_k é o número de pixels na imagem I que possuem a intensidade k.

cvCalcHist(&ImgOrigem, Histograma, 0, NULL);

Histogramas

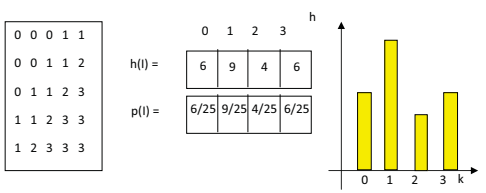
- É possível normalizar um histograma, representando os valores em termos de porcentagem:

$$p(I_k) = \frac{h(I_k)}{n} = \frac{n_k}{n}$$

Onde *n* é o número de pixels da imagem.

cvGetMinMaxHistValue(imgHist, 0, &max_value, 0, 0);
cvScale(Hist->bins, Hist->bins, ((double)cvImgHist->height)/max_value, 0);

Histogramas



À esquerda uma imagem I, ao centro o histograma da imagem em valores (h(I)) e em porcentagem (p(I)), à direita uma representação do histograma de forma gráfica.

Equalização de histogramas

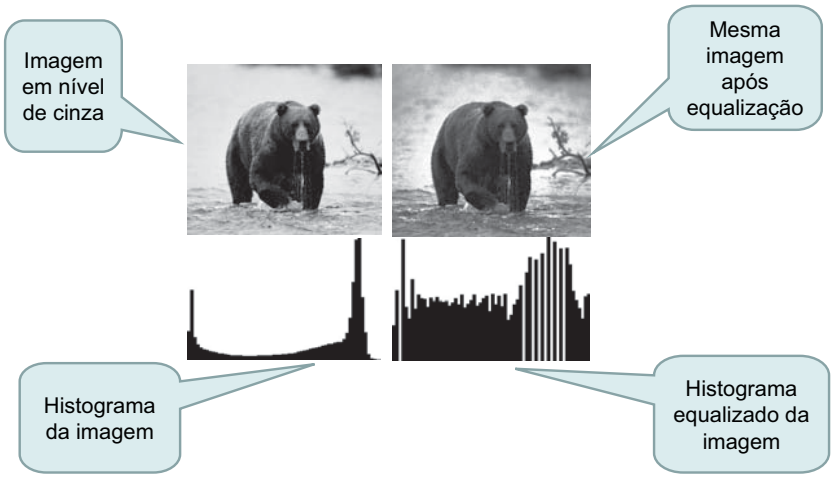
É o ajuste dos valores de intensidade de forma a melhorar o contraste em uma imagem.

$$h\_eq(k) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j$$

Onde
k é a intensidade no histograma equalizado,
L é o valor máximo de intensidade na imagem,
M e N são as dimensões da imagem e
n_j é o número de pixels na imagem com valor de intensidade igual a j.

cvEqualizeHist(ImagemOriginal, ImagemEqualizada);

Equalização de histogramas



Filtros no Domínio Espacial

- Correlação: $g(x,y) = \sum_{i=-m/2}^{m/2}\sum_{j=-n/2}^{n/2} f(x+i, y+j) * w(i,j)$

Posição (x, y) = (4, 4)

. . . . . .
. . . . . .
. . 5 7 3 . .
. . 6 4 7 . .
. . 2 5 6 . .
. . . . . .
. . . . . .

Máscara 3 x 3

1 2 1
2 4 2
1 2 1

Produto ponto a ponto

(1x5) (2x7) (1x3)
(2x6) (4x4) (2x7)
(1x2) (2x5) (1x6)

Somatória

5+14+3+12+16+14+2+10+6 =82

Resultado da Correlação

. . . . . .
. . 5 7 3 . .
. . 6 82 7 . .
. . 2 5 6 . .
. . . . . .

Filtros no Domínio Espacial

- Convolução: semelhante à correlação, porém neste caso a máscara é rotacionada de 180 graus.

$$g(x,y) = \sum_{i=-m/2}^{m/2}\sum_{j=-n/2}^{n/2} f(x-i, y-j) * w(i,j)$$

Onde w é a máscara utilizada no processo de correlação/convolução, e m e n são as dimensões da máscara.

O tipo de filtro depende dos valores da máscara - em geral, as máscaras são simétricas e, portanto, pode-se aplicar tanto a convolução como a correlação.

cvFilter2D(ImgOriginal, ImgFiltrada, filtro, cvPoint(-1,-1));

Filtros no Domínio Espacial
Estatísticos

28

- Tipos: média, mediana, moda, mínimo e máximo.
- O filtro de média, também chamado de filtro-caixa, é um filtro do tipo passa-baixa.
- O efeito de um filtro passa-baixa é de suavização da imagem e minimização dos ruídos, atenuando as transições abruptas que correspondem a frequências altas
  - porém, o efeito acaba sendo de embassamento ou borramento da imagem que acaba removendo os detalhes finos da imagem

Filtragem no Domínio Espacial
Filtros estatísticos

29

- A expressão de um filtro de média é dada por:

$$g(x,y) = \begin{cases} \dfrac{1}{mn}, & \text{se } |x| < \dfrac{m}{2} \text{ e } |y| < \dfrac{n}{2} \\ 0, & \text{caso contrario} \end{cases}$$

Onde *m* e *n* são as dimensões de uma máscara qualquer.

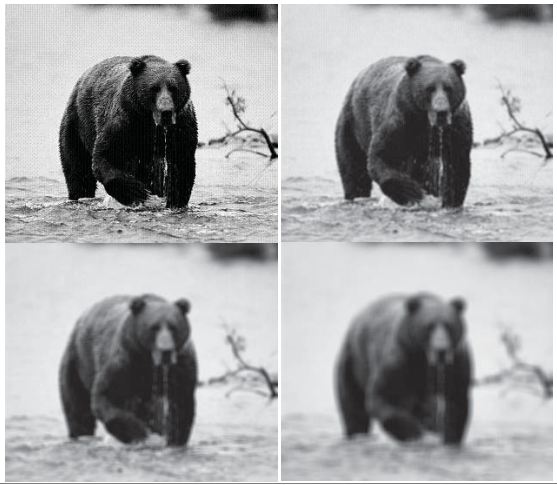
h(i,j) = 1/9*

1	1	1
1	1	1
1	1	1

Filtragem no Domínio Espacial
Filtros estatísticos

30

- Exemplos de uso de filtro de média:



Filtragem no Domínio Espacial
Filtros estatísticos

31

Filtro média com atenuação variando de acordo com a distância e orientação ao centro da máscara.

h(i,j) = 1/16*

1	2	1
2	4	2
1	2	1

Filtragem no Domínio Espacial
Filtros estatísticos

32

- No filtro da mediana os valores dos pixels são ordenados e o valor que ocupa a posição mediana dos valores é selecionado para a posição (x, y) da imagem filtrada.
  - Este filtro tende a reduzir o efeito de ruído de pulso, do tipo "*salt and peper*" pois valores pontuais raramente aparecem juntos e portanto nunca ocupam a posição mediana.

Filtragem no Domínio Espacial
Filtros estatísticos

33

- O filtro de máximo substitui o valor da posição (x, y) pelo valor máximo da máscara.
  - Este filtro tem a tendência de clarear a imagem.
- Analogamente o filtro de mínimo substitui o valor da posição (x, y) pelo valor mínimo da máscara.
  - Este filtro tem a tendência de escurecer a imagem.
- O filtro de moda seleciona para a posição (x, y) da imagem o valor que ocorre com maior frequência na máscara
  - Este tipo de filtro tende a homogeneizar os valores na imagem.

Filtragem no Domínio Espacial
Filtros estatísticos

34

- Filtros estatísticos

.	.	.	.	.	.	.	.	.
.	.	4	6	3	.	.		
.	.	6	4	5	.	.		
.	.	5	7	6	.	.		
.	.	.	.	.	.	.		

.	.	.	.	.	.	.	.	.
.	.	4	6	3	.	.		
.	.	6	?	5	.	.		
.	.	5	7	6	.	.		
.	.	.	.	.	.	.		

Máscara 3 x 3

4	6	3
6	4	5
5	7	6

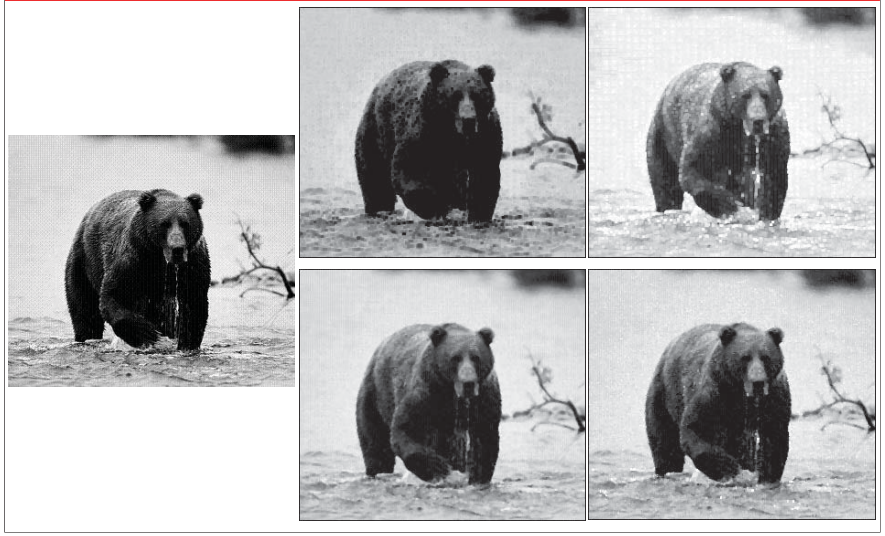
Ordenação

3 4 4 5 5 6 6 6 7

? =
- Mínimo = 3
- Maximo = 7
- Mediana = 5
- Moda = 6

Filtragem no Domínio Espacial
Filtros estatísticos

35



Filtragem no Domínio Espacial
Filtros gaussianos

- Um filtro Gaussiano tem os valores da máscara determinados a partir de uma função bidimensional Gaussiana discreta, com média igual a zero e desvio padrão σ, do tipo:

$$Gauss(x,y) = \frac{1}{2\pi\sigma^2}\exp(\frac{-(x^2+y^2)}{2\sigma^2})$$

Onde x e y são as posições na máscara e Gauss(x, y) dá o valor a ser colocado na posição (x, y) da máscara.

Filtragem no Domínio Espacial
Filtros gaussianos

- Os filtros Gaussianos são filtros de média e são utilizados para suavizar a imagem de forma ponderada e simétrica.

$$Gauss(x,y) = 1/256 * \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Máscara Gaussiana para um filtro do tipo passa-baixa.

Filtragem no Domínio Espacial
Filtros gaussianos

Filtragem no Domínio Espacial
Filtros passa-alta

- O filtro do tipo passa-alta é utilizado para realçar bordas ou regiões de interesse com transições abruptas de intensidade.
- O problema deste tipo de filtro é que ele geralmente realça também ruídos do processo de obtenção da imagem.

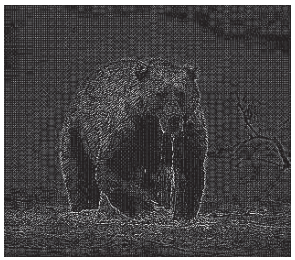
Filtragem no Domínio Espacial
Filtro passa-alta

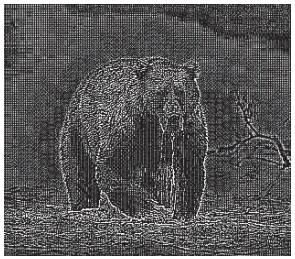
$h(i,j) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

$h(i,j) = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

Máscara para um filtro do tipo passa-alta, note que a soma dos valores dentro da máscara somam zero.

Filtragem no Domínio Espacial
Filtro passa-alta





Filtragem no Domínio Espacial
Exemplo


```
// Filtro Espacial
// criado por Mauricio Marengoni e Denise Stringhini
// versao: 1.0, em 08/2008

#include "cv.h"
#include "highgui.h"
#include <stdio.h>
int main()
{
    char name0[ ] = "imagens/ursoPretoCinzaSmall.jpg";
    IplImage *cvImagem, *cvImgFiltrada, *cvImgCinza;
    CvSize tamanhoImagem;
    CvMat *filtro;
    int lado=3;
    double kernel[ ] = { -1, -1,  -1,
                         -1,  8,  -1,
                         -1, -1,  -1};

    cvImagem = cvLoadImage( name0, -1 );
    tamanhoImagem.width=cvImagem->width;
    tamanhoImagem.height=cvImagem->height;
    cvImgFiltrada=cvCreateImage(tamanhoImagem, IPL_DEPTH_8U, 1);
    cvImgCinza=cvCreateImage(tamanhoImagem, IPL_DEPTH_8U, 1);
    cvCvtColor(cvImagem, cvImgCinza, CV_BGR2GRAY);
    //Create convolution filter
    filtro = cvCreateMatHeader( lado, lado, CV_64FC1 );
    cvSetData( filtro, kernel, lado*8 );
    //Call OpenCV convolution:
    cvFilter2D( cvImgCinza, cvImgFiltrada, filtro, cvPoint(-1,-1) );
    //salva imagem filtrada
    cvSaveImage("imagens/ursoPretoPassaAlta2.jpg",cvImgFiltrada);

    return( 0 );
}
```

Domínio de Freqüência

- É possível fazer uma troca de base em uma imagem e representá-la em termos uma soma ponderada infinita de um conjunto de senóides.
  - Mudanças rápidas na imagem são representadas por freqüências altas.
  - Mudanças suaves são representadas por frequências baixas.
- Esta mudança de base pode ser feita utilizando a transformada de Fourier.

Domínio de Freqüência

- Expressões que computam a transformada de Fourier:

Modo contínuo

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}$$

Modo discreto

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$$

A expressão discreta é a utilizada na DFT (*Discrete Fourier Transform*) que geralmente é usada na implementação em computador da transformada de Fourier.

Domínio de Freqüência

- Pode ser mostrado que o processo de convolução no domínio espacial corresponde a multiplicação de duas expressões no domínio de frequência, isto é:

$$g(x,y) = f(x,y) ** w(i,j) \Leftrightarrow G(u,v) = F(u,v) * W(r,s)$$

Onde ** indica a convolução entre a imagem f(x,y) com a máscara w(i,j)
G(u,v) é a transformada de Fourier de g(x,y), que é o produto de F(u,v) por W(r,s)
F(u,v) é a transformada de Fourier de f(x,y)
W(r,s) é a transformada de Fourier de w(i,j)

Domínio de Freqüência
Processo de filtragem

1. Dada uma imagem f(x,y) de tamanho M x N, obter uma imagem com entorno fp(x,y) de tamanho P x Q onde P = 2M e Q = 2N preenchida com zeros no entorno.
2. Multiplique fp(x,y) por $(-1)^{(x+y)}$ para centralizar a transformada.
3. Determinar a DFT da imagem obtida em b.
4. Criar um filtro simétrico H(u,v) de tamanho P x Q com centro em (P/2,Q/2)

Domínio de Freqüência
Processo de filtragem (cont.)

5. Fazer o produto H(u,v)*F(u,v) obtido na etapa 3.
6. Obter a imagem no domínio de espaço fazendo o inverso da transformada de Fourier gp(x,y).
7. Multiplicar gp(x,y) por (-1)(x+y)
8. Remover o entorno da imagem obtendo então g(x,y), que é a imagem filtrada.

Domínio de Freqüência

48

- A expressão que determina o inverso da transformada de Fourier discreta é dada por:

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)e^{j2\pi(ux/M+vy/N)}$$

Portanto, definir um filtro no domínio de frequência corresponde a encontrar uma máscara para ser utilizada em conjunto com a imagem transformada e assim obter a imagem filtrada desejada.

Domínio de Freqüência
Filtros Passa-baixa

49

- Um filtro passa-baixa ideal é dado pela expressão:

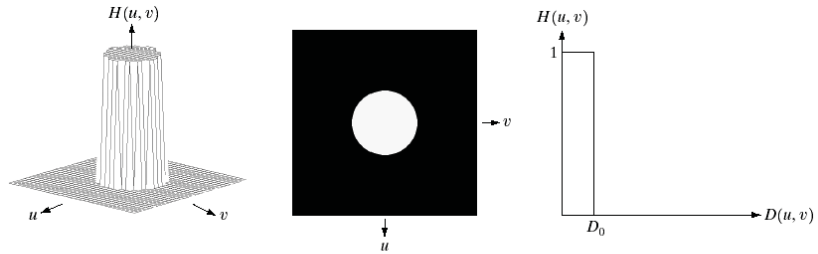
$$H(u,v) = \begin{cases} 1, & \text{se } F(u,v) < F_0 \\ 0, & \text{caso contrario} \end{cases}$$

Onde F_0 é uma frequência de corte.

Este filtro faz um corte abrupto em uma certa frequência, por isso o nome de filtro ideal.

Domínio de Freqüência
Filtros Passa-baixa

50



a b c

FIGURE 4.10 (a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

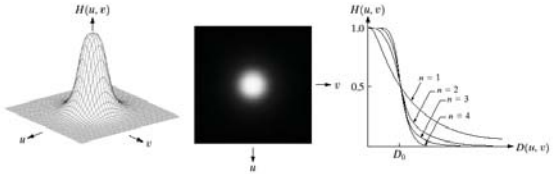
Domínio de Freqüência
Filtros Passa-baixa

51

- Outro filtro do tipo passa-baixa é o filtro de Butterworth cuja expressão é dada por:

$$H(u,v) = \sqrt{\frac{1}{1+(F(u,v)/F_0)^{2n}}}$$

Onde n define a ordem do filtro de Butterworth. Este valor indica a forma de atenuação da freqüência a partir da origem do filtro.



a b c

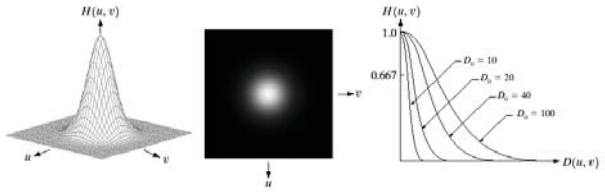
FIGURE 4.14 (a) Perspective plot of a Butterworth lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4.

Domínio de Freqüência
Filtros Passa-baixa

- O filtro Gaussiano no domínio de freqüência é dado pela seguinte expressão:

$$H(u,v) = e^{-F^2(u,v)/2\sigma^2}$$

O filtro Gaussiano obtém uma atenuação menos suave que o filtro de Butterworth para a mesma frequência de corte.



a b c

FIGURE 4.17 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

Domínio de Freqüência
Filtros Passa-alta

- Filtro passa-alta ideal:

$$H(u,v) = \begin{cases} 1, & \text{se } F(u,v) > F_0 \\ 0, & \text{caso contrario} \end{cases}$$

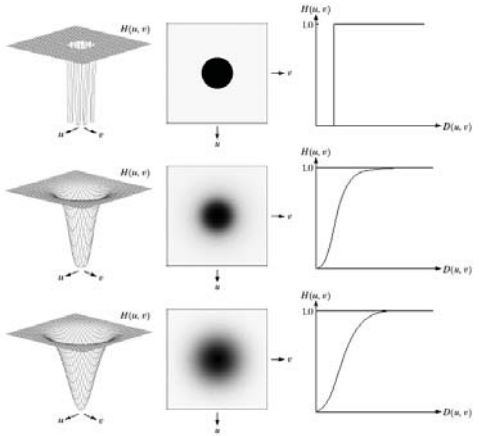
- Filtro passa-alta de Butterworth:

$$H(u,v) = \frac{1}{\sqrt{1 + (F_0 / F(u,v))^{2n}}}$$

- Filtro passa-alta Gaussiano:

$$H(u,v) = 1 - e^{-F^2(u,v)/2\sigma^2}$$

Domínio de Freqüência
Filtros Passa-alta



a b c
d e f
g h i

FIGURE 4.22 Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

Segmentação de Imagens

- Segmentação :
  - Partição da imagem em regiões que possuem algum tipo de semelhança.
  - Etapa inicial no processo de identificação de objetos numa imagem.
  - Existem diversas técnicas para segmentar uma imagem.

Segmentação de Imagens

56

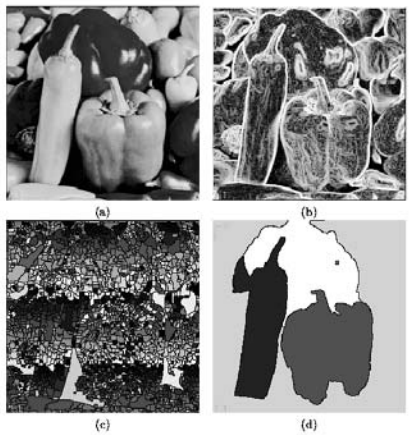


Figure 5.51: *Watershed segmentation: (a) original; (b) gradient image, 3×3 Sobel edge detection, histogram equalized; (c) raw watershed segmentation; (d) watershed segmentation using region markers to control oversegmentation. Courtesy W. Higgins, Penn State University.*

Segmentação de Imagens

57

- Técnicas de Segmentação :
  - Detecção de Bordas
  - Por Corte
  - Baseada em Regiões
  - Limiarização
  - Movimento

Reconhecimento de Objetos

58

- O reconhecimento de objetos ou padrões em uma imagem é um dos principais objetivos de um processo de visão computacional.
- Existem diversas técnicas para fazer reconhecimento de padrões, geralmente agrupadas como:
  - Estruturais
  - Baseados em teoria da decisão

Reconhecimento de Objetos

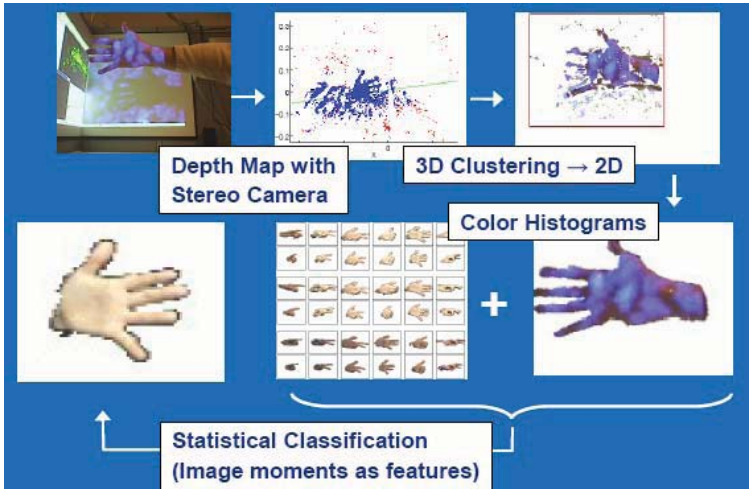
59

Reconhecimento de Objetos

60

Reconhecimento de Objetos

61



Reconhecimento de Objetos

62

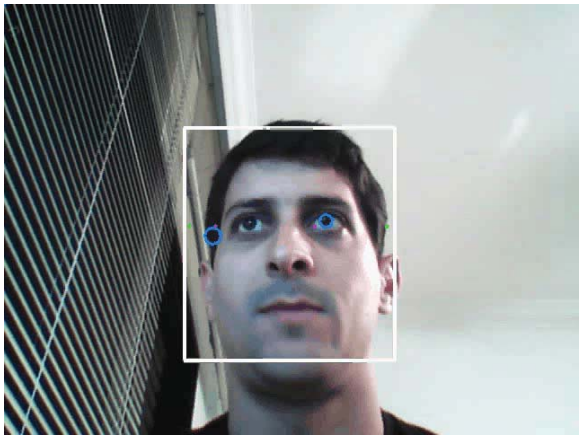
- Reconhecimento de objetos baseado em teoria da decisão:
  - Classificador de distância mínima.
  - Correlação
  - Classificadores Estatísticos
  - Redes Neurais
- Reconhecimento de objetos baseado em estruturas:
  - Baseado em Forma.

Rastreamento de Objetos

63

- Rastreamento de Objetos é uma operação utilizada para, uma vez identificado um objeto, acompanhá-lo em uma seqüência de imagens.
- Existem diversas aplicações para o rastreamento:
  - Comportamentos
  - Verificação de visada
  - Segurança

Rastreamento de Objetos



Exemplo: *camshiftdemo.c*

- CamShift (*Continuously Adaptive Mean-SHIFT*) é um algoritmo desenvolvido para o rastreamento de cor, possibilitando também o rastreamento de faces.
- É baseado numa técnica estatística onde se busca o pico entre distribuições de probabilidade em gradientes de densidade.
- Esta técnica é chamada de “média por deslocamento” (*mean shift*) e foi adaptada no CamShift para tratar a mudança dinâmica das distribuições de probabilidade das cores numa seqüência de vídeo.

Exemplo: *camshiftdemo.c*

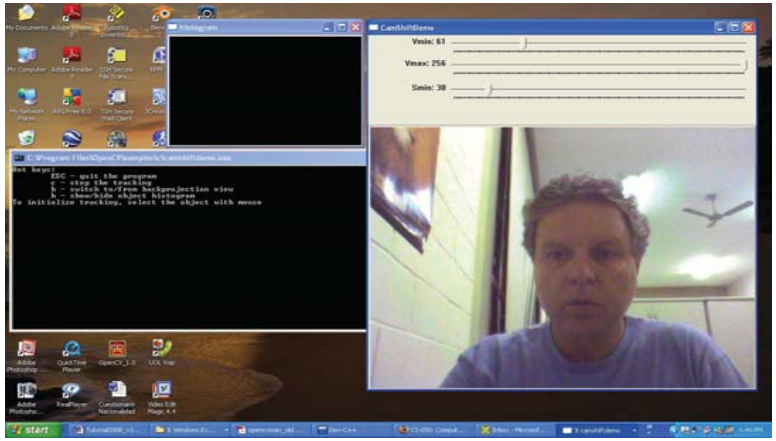
- Para cada frame, a imagem (*raw*) é convertida para outra de distribuição de probabilidade de cor através de um modelo de histograma da cor da pele (no caso de rastreamento de faces).
- O centro e o tamanho da face que se quer rastrear são encontrados através do CamShift operando na imagem de probabilidade de cores.
- O tamanho e a localização corrente da face são informados e usados para definir o tamanho e a localização da janela de busca da próxima imagem de vídeo.

Exemplo: *camshiftdemo.c*
Inicialização do programa


```
int main( int argc, char** argv ){
    CvCapture* capture = 0;
    if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 && isdigit(argv[1][0])))
        //inicializa a captura de vídeo a partir de uma câmera
        capture = cvCaptureFromCAM(argc == 2 ? argv[1][0] - '0' : 0 );
    else if( argc == 2 )
        //inicializa a captura de vídeo a partir de um arquivo de vídeo
        capture = cvCaptureFromAVI( argv[1] );
    if( !capture ){
        fprintf(stderr,"Could not initialize capturing...\n");
        return -1;
    }
    printf( "Hot keys: \n" //…);

    cvNamedWindow( "Histogram", 1 ); //cria a janela para o histograma
    cvNamedWindow( "CamShiftDemo", 1 ); //cria a janela para o rastreamento

    //associa a função on_mouse à janela de rastreamento
    cvSetMouseCallback( "CamShiftDemo", on_mouse, 0 );

   //cria os 3 sliders para ajuste dos parâmetros (janela CamShiftDemo)
    cvCreateTrackbar( "Vmin", "CamShiftDemo", &vmin, 256, 0 );
    cvCreateTrackbar( "Vmax", "CamShiftDemo", &vmax, 256, 0 );
    cvCreateTrackbar( "Smin", "CamShiftDemo", &smin, 256, 0 );
```

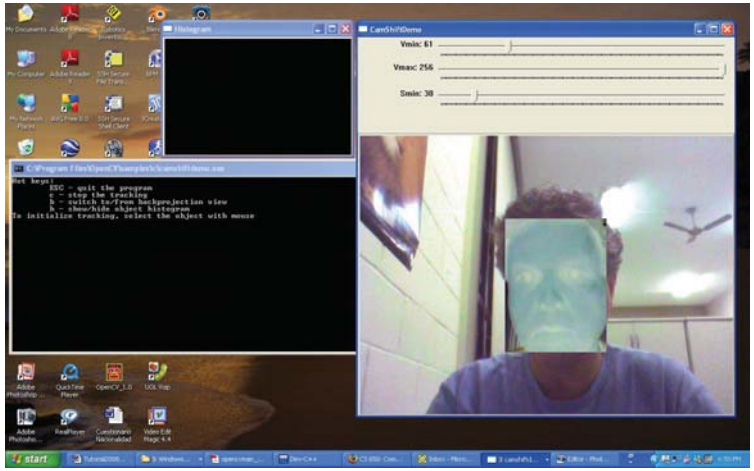
Exemplo: *camshiftdemo.c*

Janelas criadas



Exemplo: *camshiftdemo.c*

Definição do objeto (função *on_mouse*)



Exemplo: *camshiftdemo.c*

Trechos selecionados

```
cvInRangeS( hsv, cvScalar(0,smin,MIN(_vmin,_vmax),0),
            cvScalar(180,256,MAX(_vmin,_vmax),0), mask );
cvSplit( hsv, hue, 0, 0, 0 );
```

- `cvInRangeS`: verifica se os elementos em *hsv* (previamente convertida a partir da imagem RGB obtida do frame original) estão entre os dois escalares indicados nos parâmetros 2 e 3. O parâmetro *mask* é a imagem destino que armazenará os pixels que interessarão no cálculo do histograma.
- `cvSplit`: retira um dos canais e armazena na imagem de *hue* (matiz), que será usada no cálculo do histograma e da sua projeção de fundo (retro-projeção ou *back projection*).

Exemplo: *camshiftdemo.c*

Trechos selecionados

```
cvCalcBackProject( &hue, backproject, hist );
cvAnd( backproject, mask, backproject, 0 );
cvCamShift( backproject, track_window,
            cvTermCriteria(CV_TERMCRIT_EPS | CV_TERMCRIT_ITER,
            0, 1), &track_comp, &track_box );
```

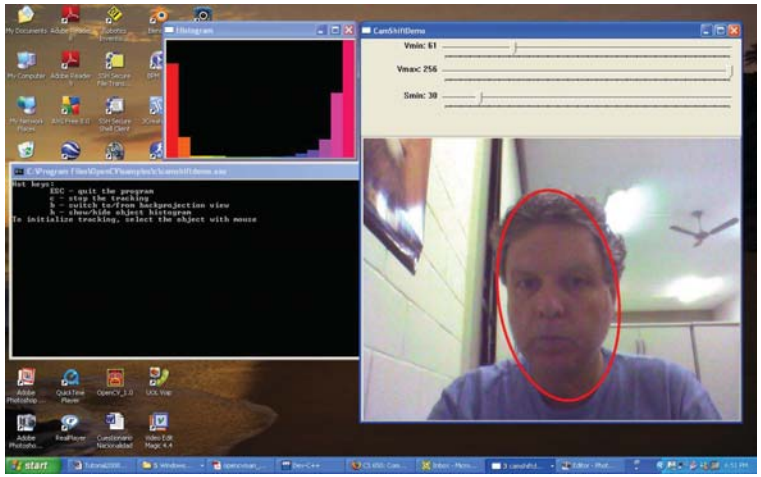
- `cvCalcBackProject`: calcula a projeção de fundo (retro-projeção) do histograma baseado na imagem de matiz (*hue*).
- `cvAnd`: calcula um E lógico entre a imagem de fundo do histograma e a máscara calculada anteriormente (que possui os pixels de interesse) - o resultado é novamente armazenado na imagem de fundo do histograma (*backproject*).
- `cvCamShift`: chama o algoritmo CamShift para buscar o centro, o tamanho e a orientação do objeto sendo rastreado e armazenar em *track_comp.*

Exemplo: *camshiftdemo.c*
Trechos selecionados

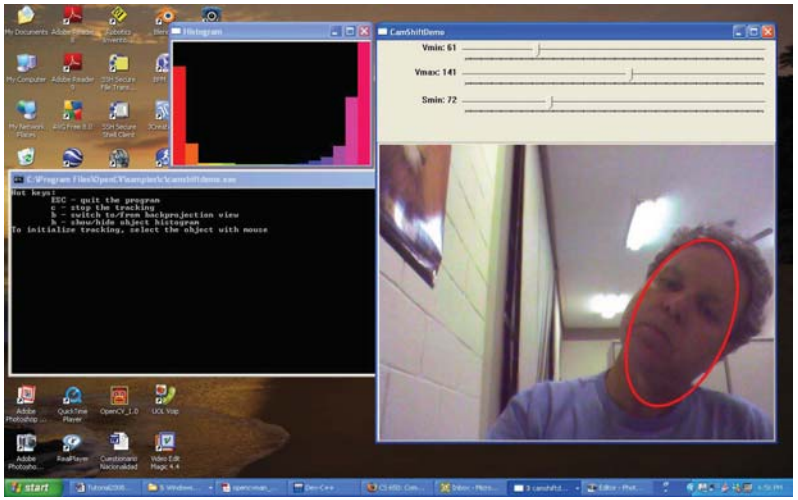

```
track_window = track_comp.rect;
if( backproject_mode )
     cvCvtColor( backproject, image, CV_GRAY2BGR );
if( image->origin )
     track_box.angle = -track_box.angle;
     cvEllipseBox(image, track_box, CV_RGB(255,0,0), 3,CV_AA, 0);
```

- `track_window` recebe a área retangular onde está inserido o objeto rastreado e uma elipse é desenhada ao redor do objeto.

Exemplo: *camshiftdemo.c*
Objeto detectado



Exemplo: *camshiftdemo.c*
Objeto rastreado



Ajuste do ambiente do OpenCV

Windows

76

Ajustar a PATH para a biblioteca:

- Em “Meu Computador” use o botão direito do mouse e selecione Propriedades.
- Clique na aba “Avançado” e após no botão de “Variáveis do Ambiente” na parte inferior da janela (no Vista ainda será necessário seguir o link “*Advanced System Settings*” antes deste passo).

Windows

77

Ajustar a PATH para a biblioteca *(cont.)*:

- Nas “Variáveis do usuário” está a PATH, edite-a para acrescentar o local onde a biblioteca OpenCV foi instalada
  - o caminho padrão é

    `C:\Arquivos de Programa\OpenCV\bin`
- Acrescente também o caminho para a biblioteca de interface
  - o caminho padrão é

    `C:\ Arquivos de Programa\OpenCV\otherlibs\highgui`
- Lembre-se de separá-las utilizando ”;”.

OBS: é possível que a PATH do OpenCV seja atualizada automaticamente durante a instalação, mas é necessário verificar.

Windows

78

Visual Studio 2005

- Abrir o ambiente do Visual Studio 2005.
- Abrir o arquivo *opencv.sln* na pasta *_make* da instalação (C:\Arquivos de Programa\OpenCV_make). Isto deve carregar toda a solução do OpenCV no ambiente do Visual Studio 2005.
- Selecionar a opção “Build” no menu e clicar em “*build-solution*” para compilar a biblioteca. Esta compilação pode levar algum tempo.

Windows

79

DevCPP

- Abrir o Bloodshed Dev-C++ (*versão testada: 4.9.9.2*).
- Para ligar (*link*) os programas desenvolvidos no Dev C++ com as bibliotecas do OpenCV:
  - Clicar em “*Tools*” e selecionar “*Compiler Options*”. O sistema deve abrir uma janela com a aba “*Compiler*” selecionada.
  - Marcar o *box* “*Add these commands to the linker command line*” e copiar os seguintes comandos:

    `-lhighgui -lcv -lcxcore -lcvaux -lcvcam`

Windows

DevCPP *(cont.)*

- Na aba "*Directories"*, em "*Binaries*" adicionar o caminho: C:\Arquivos de Programa\OpenCV\bin.
- Em "*C Includes*" adicionar os caminhos:

```
C:\Arquivos de
Programa\OpenCV\cxcore\include;C:\Arquivos de
Programa\OpenCV\cv\include;C:\Arquivos de
Programa\OpenCV\otherlibs\highgui;C:\Arquivos de
Programa\OpenCV\cvaux\include; C:\Arquivos de
Programa\OpenCV\otherlibs\cvcam\include
```

- Em "*C++ Includes*" adicionar os mesmos caminhos acima.
- Em "*Libraries*" adicionar o caminho:

```
C:\Arquivos de Programa\OpenCV\lib
```

Windows

Testar a instalação no Windows

- Abrir o ambiente de trabalho escolhido e carregar o arquivo: C:\Arquivos de Programa\OpenCV\samples\c\contours.c.
- Criar um novo arquivo de programa em branco: teste.cpp.
- Copiar e colar todo o arquivo contours.c para o arquivo em branco teste.cpp, salve o arquivo teste.cpp.
- Compilar e executar o programa teste.cpp.
  - Este programa analisa uma imagem sintética com faces e extrai regiões diferentes da imagem. Pode-se alterar parâmetros ou linhas do código e ver os resultados. (Não esquecer de recompilar o programa após as alterações).

Linux e g++

- Entrar com *login* de super-usuário.
- Ir ao diretório onde foi feito o *download* do arquivo *tar.gz* do OpenCV.
- Na janela aberta, digitar os seguintes comandos:

```
tar -xzf opencv-0.9.7.tar.gz
cd opencv-0.9.7
./configure && make && make install
```

Linux e g++

- Fazer os ajustes para o G++:
  - Ir ao diretório "*home*" e abrir o arquivo *.bashrc* (para o ambiente *bash*).
  - Acrescentar a seguinte linha ao arquivo:

```
alias gcv="g++ -I/usr/local/include/opencv -lcv
-lcxcore -lcvaux -lhighgui"
```

Linux e g++

Testar a instalação no Linux

- Ir ao diretório onde está instalado o OpenCV.
- Procurar o sub-diretório "*samples*" e entrar no sub-diretório "*c*".
- Digitar a linha de comando abaixo que deve executar o programa *contours.c*:

```
gcv contours.c && ./a.out
```