



# Criação de imagens e vídeos 3D com OpenCV

Matheus Ricardo Uihara Zingarelli  
(zinga@icmc.usp.br)



# Apresentação

- Conhecimento Técnico
  - C / C++
  - Básico
    - printf()
    - struct
    - loop
    - Argumentos de linha de comando
    - Funções
    - Ponteiros



# Apresentação

- Verificar instalação
  - OpenCV 2.1 + CodeBlocks



# Apresentação

- Verificar instalação
  - OpenCV 2.1 + CodeBlocks
  - [www.icmc.usp.br/~zinga/SemComp](http://www.icmc.usp.br/~zinga/SemComp)
  - Baixar: “Código Demo”



# Página de desambiguação

- O que esperam aprender?



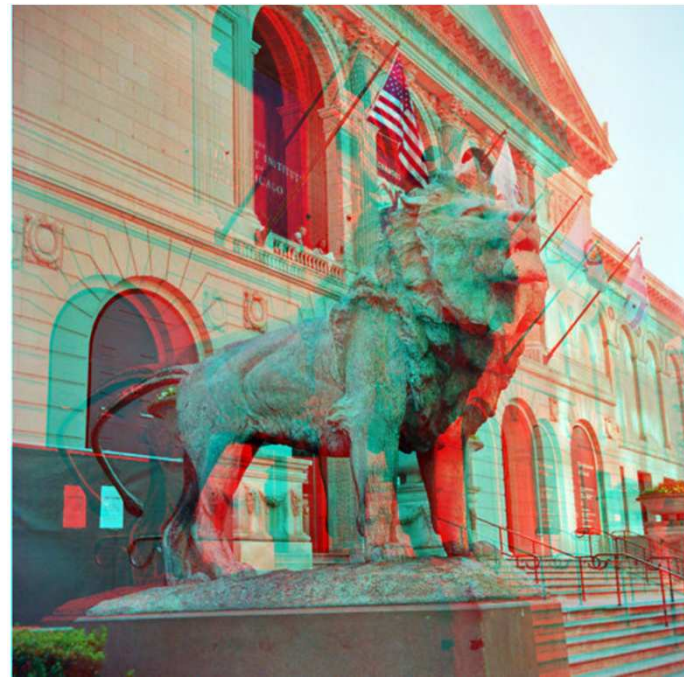
# Página de desambiguação

- O que esperam aprender?



# Página de desambiguação

- O que esperam aprender?

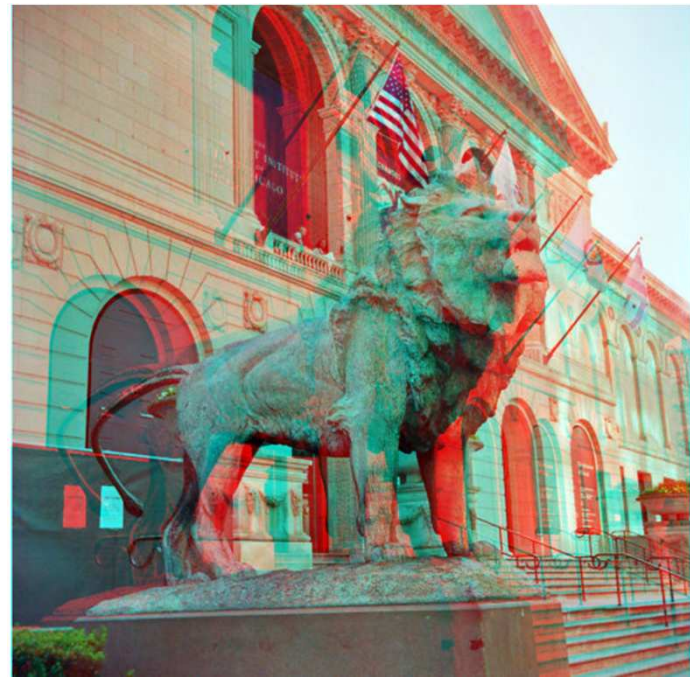


# Página de desambiguação

- O que esperam aprender?



?





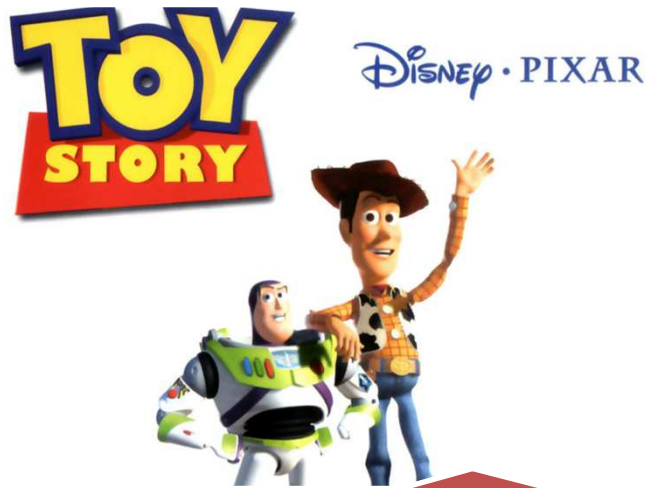
# Página de desambiguação

- O que esperam aprender?



# Página de desambiguação

- O que esperam aprender?



Animação 3D



3D estereoscópico

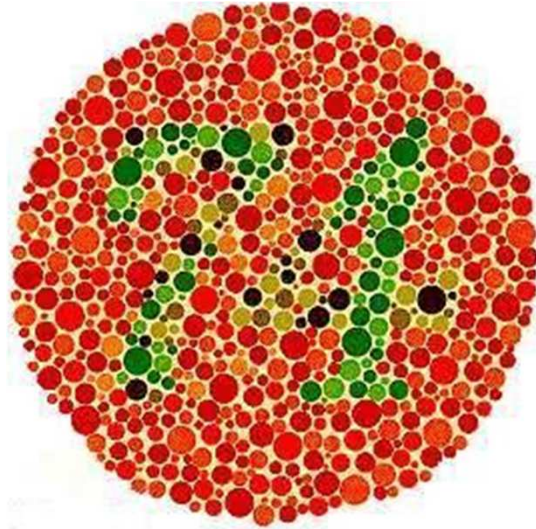
# 3D estereoscópico

- Nem todos conseguem visualizar...



# 3D estereoscópico

- Nem todos conseguem visualizar...



# Agenda

- OpenCV 101
- Imagens: funções básicas
- Fundamentos 3D estereoscópico
- Criação de imagem anáglifa
- Vídeos: funções básicas
- Criação de vídeo anáglifo
- Material de referência



Começando...

# OPENCV 101



# OpenCV

- Material baseado no Livro de [Bradski & Kaehler](#)
- <http://www.amazon.com/Learning-OpenCV-Computer-Vision-Library/dp/0596516134>



# OpenCV

- O que é?
- Para que serve?
- Quem utiliza?
- Como programo?





# OpenCV

- O que é?

***“OpenCV (Open Source Computer Vision) is a library of programming functions for real time computer vision.” (OpenCV Wiki)***

- Para que serve?
- Quem utiliza?
- Como programo?



# OpenCV – O que é?

- Milhares de algoritmos otimizados visando eficiência
- Rapidez na criação de aplicações de visão computacional → reuso
- Origem nos laboratórios da Intel
- *Open Source* sob a licença BSD



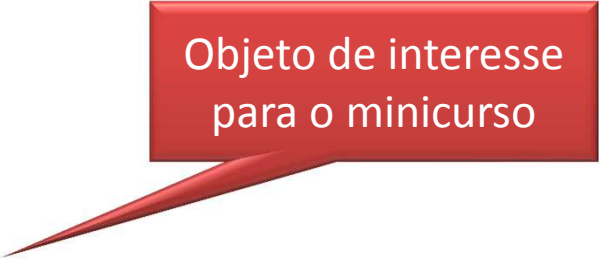
# OpenCV

- O que é?
- Para que serve?
  - Muita coisa
    - Processamento de imagens, calibração de câmeras, monitoramento, [rastreamento](#), [reconhecimento facial/gestos](#), análise de imagens médicas, [segmentação](#), [Kinect](#), ...
- Quem utiliza?
- Como programo?



# OpenCV

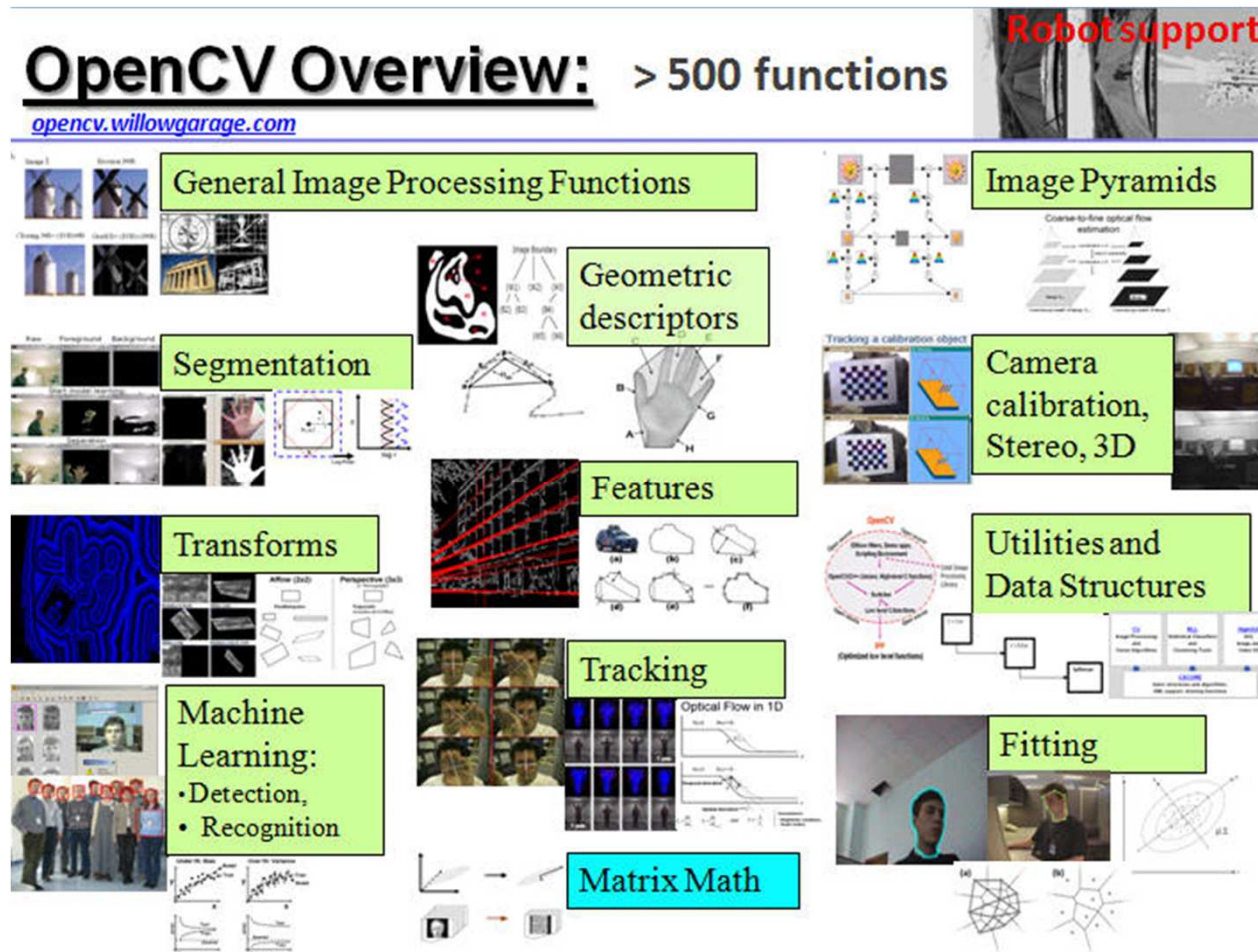
- O que é?
- Para que serve?
  - Muita coisa
    - **Processamento de imagens**, calibração de câmeras, monitoramento, [rastreamento](#), [reconhecimento facial/gestos](#), análise de imagens médicas, [segmentação](#), [Kinect](#), ...
- Quem utiliza?
- Como programo?



Objeto de interesse  
para o minicurso



# OpenCV – Para que serve?



Algumas funções do OpenCV (OpenCV - Wiki)



# OpenCV

- O que é?
- Para que serve?
- Quem utiliza?
  - Muitas empresas
    - Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, Applied Minds, VideoSurf, Zeitera,...
  - Governos
    - *Green Dam da China*

*“The purported intent of the Green Dam software is to filter harmful online text and image content in order to prevent the effects of this information on youth and promote a healthy and harmonious Internet environment”*

*(OpenNet Bulletin)*
- Como programo?



# OpenCV

- O que é?
- Para que serve?
- Quem utiliza?
- Como programo?
  - C, C++, Python
  - Em desenvolvimento: Java, Ruby, Matlab e outros times paralelos
  - Windows, Linux, Android, Mac



# OpenCV – Como Programo?

- Download do OpenCV
  - <http://sourceforge.net/projects/opencvlibrary>
- Wiki
  - Guias para compilar/instalar
  - <http://opencv.willowgarage.com/wiki/FullOpenCVWiki>
- Documentação
  - Referência para funções
  - <http://opencv.itseez.com/>
- Livros
- Instalação para o minicurso
  - OpenCV 2.1
  - Codeblocks 10.05
  - Tutorial <http://xcelovers.wordpress.com/2011/02/03/tutorial-integrating-codeblocks-and-opencv-2-1-0/>





Dúvidas até aqui?

**CHECKPOINT**





Mão na massa

# IMAGENS: FUNÇÕES BÁSICAS

# Operações

- Abrir imagem
- Ler imagem
- Processamento
- Criar imagem



# HighGUI

- Toolkit para funções de *user interface*
  - Estrutura para trabalhar com imagens e vídeos
  - Criação de janelas, *sliders*, botões
  - Tratamento de eventos de mouse, teclado

```
include "highgui.h"
```



# cvLoadImage()

- Função para abrir imagens em disco
- Aloca memória e armazena a imagem em uma estrutura apropriada
- BMP, DIB, JPEG, JPE, PNG, PBM, PGM, PPM, SR, RAS e TIFF → **não suporta gif!**

```
IplImage* cvLoadImage(  
    const char* filename,  
    int iscolor = CV_LOAD_IMAGE_COLOR  
);
```

Nome do arquivo  
com extensão

CV\_LOAD\_IMAGE\_COLOR = colorida  
CV\_LOAD\_IMAGE\_GRAYSCALE = tons de cinza



# cvNamedWindow()

- Cria janela para mostrar imagens

```
int cvNamedWindow(  
    const char* name,  
    int flags = CV_WINDOW_AUTOSIZE  
);
```

Nome da janela. Usado como handler para operações na janela

CV\_WINDOW\_AUTOSIZE = janela do tamanho da imagem  
0 = permite que usuário redefina o tamanho da janela



# cvShowImage()

- Ligação entre a imagem e a janela

```
void cvShowImage(  
    const char*   name,  
    const CvArr*  image  
);
```

Handler da janela

Nome da variável que aponta para a imagem carregada em cvLoadImage()



# cvWaitKey()

- Aguarda interação do usuário com o teclado

```
int cvWaitKey(  
    int delay = 0  
);
```

Tempo de espera em ms.  
Default 0 aguarda para sempre





# Limpeza

- Liberação dos ponteiros que carregaram as estruturas

```
void cvReleaseImage( IplImage** img );
```

```
void cvDestroyWindow( const char* name );
```



# Exercício 01

- Abrir uma imagem e mostrar na tela



# Operações

- Abrir imagem ✓
- Ler imagem
- Processamento
- Criar imagem



Dúvidas até aqui?

**CHECKPOINT**



# IplImage

- Estrutura para tratamento de imagens

```
typedef struct _IplImage {
    int                nSize;
    int                ID;
    int                nChannels;
    int                alphaChannel;
    int                depth;
    char               colorModel[4];
    char               channelSeq[4];
    int                dataOrder;
    int                origin;
    int                align;
    int                width;
    int                height;
    struct _IplROI*     roi;
    struct _IplImage*   maskROI;
    void*              imageId;
    struct _IplTileInfo* tileInfo;
    int                imageSize;
    char*              imageData;
    int                widthStep;
    int                BorderMode[4];
    int                BorderConst[4];
    char*              imageDataOrigin;
} IplImage;
```



# IplImage

- Estrutura para tratamento de imagens

```
typedef struct _IplImage {  
    int             nSize;  
    int             ID;  
    int             nChannels;  
    int             alphaChannel;  
    int             depth;  
    char            colorModel[4];  
    char            channelSeq[4];  
    int             dataOrder;  
    int             origin;  
    int             align;  
    int             width;  
    int             height;  
    struct _IplROI*  roi;  
    struct _IplImage* maskROI;  
    void*           imageId;  
    struct _IplTileInfo* tileInfo;  
    int             imageSize;  
    char*           imageData;  
    int             widthStep;  
    int             BorderMode[4];  
    int             BorderConst[4];  
    char*           imageDataOrigin;  
} IplImage;
```

Número de canais



# IplImage

- Estrutura para tratamento de imagens

```
typedef struct _IplImage {  
    int                nSize;  
    int                ID;  
    int                nChannels;  
    int                alphaChannel;  
    int                depth;  
    char               colorModel[4];  
    char               channelSeq[4];  
    int                dataOrder;  
    int                origin;  
    int                align;  
    int                width;  
    int                height;  
    struct _IplROI*     roi;  
    struct _IplImage*   maskROI;  
    void*              imageId;  
    struct _IplTileInfo* tileInfo;  
    int                imageSize;  
    char*              imageData;  
    int                widthStep;  
    int                BorderMode[4];  
    int                BorderConst[4];  
    char*              imageDataOrigin;  
} IplImage;
```

Profundidade do  
pixel (bits)



# IplImage

- Estrutura para tratamento de imagens

```
typedef struct _IplImage {  
    int                nSize;  
    int                ID;  
    int                nChannels;  
    int                alphaChannel;  
    int                depth;  
    char               colorModel[4];  
    char               channelSeq[4];  
    int                dataOrder;  
    int                origin;  
    int                align;  
    int                width;  
    int                height;  
    struct _IplROI*    roi;  
    struct _IplImage*  maskROI;  
    void*              imageId;  
    struct _IplTileInfo* tileInfo;  
    int                imageSize;  
    char*              imageData;  
    int                widthStep;  
    int                BorderMode[4];  
    int                BorderConst[4];  
    char*              imageDataOrigin;  
} IplImage;
```

Ponteiro para a primeira  
linha de dados da imagem





# IplImage

- Estrutura para tratamento de imagens

```
typedef struct _IplImage {  
    int                nSize;  
    int                ID;  
    int                nChannels;  
    int                alphaChannel;  
    int                depth;  
    char               colorModel[4];  
    char               channelSeq[4];  
    int                dataOrder;  
    int                origin;  
    int                align;  
    int                width;  
    int                height;  
    struct _IplROI*    roi;  
    struct _IplImage*  maskROI;  
    void*              imageId;  
    struct _IplTileInfo* tileInfo;  
    int                imageSize;  
    char*              imageData;  
    int                widthStep;  
    int                BorderMode[4];  
    int                BorderConst[4];  
    char*              imageDataOrigin;  
} IplImage;
```

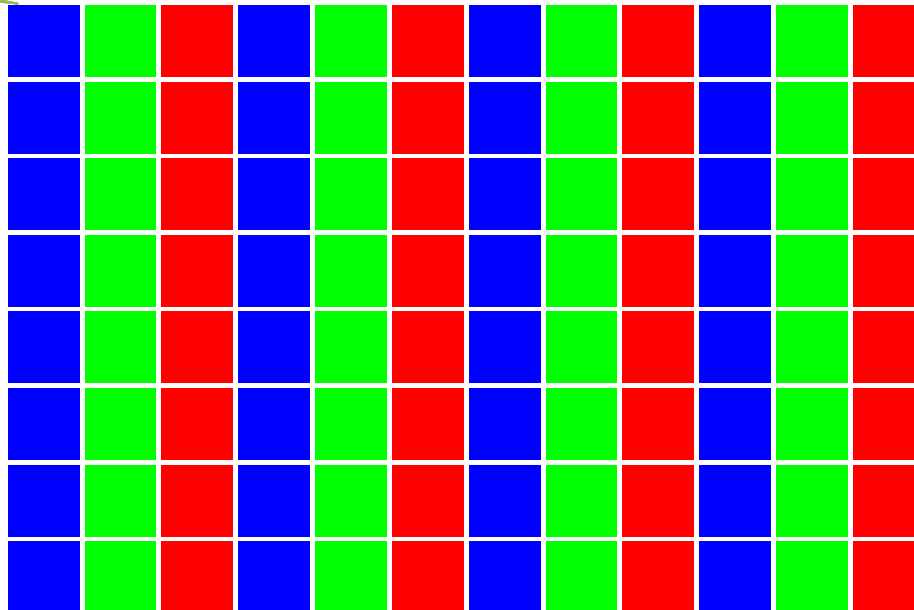
Qtde de bytes entre pontos  
situados em uma mesma  
coluna, em linhas diferentes



# Leitura

- Apenas uma matriz
- Sequência de pixels B – G – R

Início da leitura (0,0)



# Leitura

- Diferentes jeitos de se fazer a leitura de dados de uma imagem
- Nosso jeito: leitura horizontal

```
for( int row = 0; row < img->height; row++ ){
    uchar* ptr = (uchar*) ( img->imageData + row * img->widthStep );
    for( int col = 0; col < img->width; col++ ) {
        printf( "%d ", ptr[3*col] );    //acessa componente azul B
        printf( "%d ", ptr[3*col+1] ); //acessa componente verde G
        printf( "%d\n", ptr[3*col+2] ); //acessa componente vermelha R
    }
}
```



# Exercício 02

- Abrir uma imagem e imprimir dados de cor RGB na tela



# Operações

- Abrir imagem ✓
- Ler imagem ✓
- Processamento
- Criar imagem



Dúvidas até aqui?

**CHECKPOINT**



# CV

- Funções de processamento de imagens, análise de dados de imagens, reconhecimento de padrões, calibração de câmera, etc.


```
include "cv.h"
```



# cvCvtColor()

- Conversão de espaço de cores
- Imagens devem possuir mesmo número de canais e tipo de dados

```
void cvCvtColor(  
    const CvArr* src,  
    CvArr*      dst,  
    int         code  
);
```



CV\_BGR2GRAY,  
CV\_BGR2HSV,  
CV\_BGR2YCrCb,  
...





## Exercício 03

- Abrir imagem
- Realizar alguma conversão na imagem
- Mostrar a imagem original e a convertida em janelas diferentes

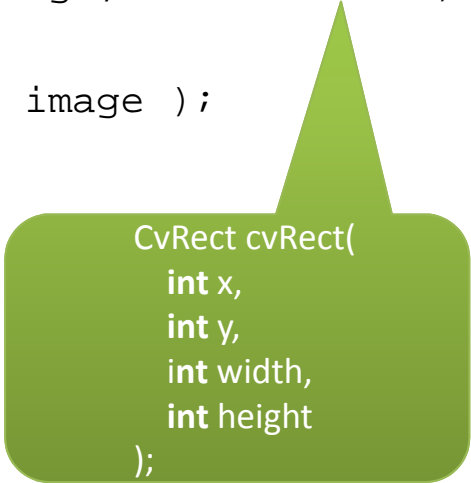


# Region Of Interest (ROI)

- “Máscara”
  - Processamento de partes específicas de uma imagem
  - Região retangular

```
void cvSetImageROI( IplImage* image, CvRect rect );
```

```
void cvResetImageROI( IplImage* image );
```



```
CvRect cvRect(  
    int x,  
    int y,  
    int width,  
    int height  
);
```



# Exercício 04

- Abrir imagem
- Cortá-la ao meio (verticalmente)
- Exibir cada metade em uma janela diferente



# Outros tipos de processamento

- Suavização
- Realce
- Redimensionamento
- ... (ver Cap. 5 Bradski & Kaehler)



# Operações

- Abrir imagem ✓
- Ler imagem ✓
- Processamento ✓
- Criar imagem



Dúvidas até aqui?

**CHECKPOINT**



# cvCreateImage()

- Criação de um container para uma nova imagem

```
IplImage* cvCreateImage(  
    CvSize size,  
    int depth,  
    int channels  
);
```

```
CvSize cvGetSize(  
    const CvArr* arr  
);  
  
CvSize cvSize(  
    int width,  
    int height  
);
```



# cvCreateImage()

- Criação de um container para uma nova imagem

```
IplImage* cvCreateImage(  
    CvSize size,  
    int depth,  
    int channels  
);
```

IPL\_DEPTH\_8U  
IPL\_DEPTH\_8S  
IPL\_DEPTH\_16U  
IPL\_DEPTH\_16S  
IPL\_DEPTH\_32S  
IPL\_DEPTH\_32F  
IPL\_DEPTH\_64F





# cvCreateImage()

- Criação de um container para uma nova imagem

```
IplImage* cvCreateImage(  
    CvSize size,  
    int depth,  
    int channels  
);
```

Ou simplesmente copie de outra imagem.

`img->depth`



# cvCreateImage()

- Criação de um container para uma nova imagem

```
IplImage* cvCreateImage(  
    CvSize size,  
    int depth,  
    int channels  
);
```

O mesmo vale para o número de canais.

`img->nChannels`



# cvSaveImage()

- Cria arquivo em disco

```
int cvSaveImage(  
    const char* filename,  
    const CvArr* image  
);
```

1 – Sucesso  
2 – Erro

Com a extensão  
no final!



# Exercício 05

- Abrir imagem **img1**
- Criar uma nova imagem **img2**
  - Mesmo tamanho
  - Apenas 1 canal
  - Mesma profundidade de pixel
- Converter imagem **img1** para tons de cinza e armazenar o resultado em **img2**.
- Salvar **img2**.



# Exercício 06

- Abrir imagem **img1**
- Criar duas novas imagens **img2** e **img3**
  - **Metade da largura** de **img1**
  - Mesmo número de canais
  - Mesma profundidade de pixel
- Cortar **img1** ao meio (verticalmente) e armazenar cada metade em **img2** e **img3**.
- Salvar **img2** e **img3**.



# Operações

- Abrir imagem ✓
- Ler imagem ✓
- Processamento ✓
- Criar imagem ✓



Dúvidas até aqui?

**CHECKPOINT**





“Aqueles óculos de papel celofane...”

# FUNDAMENTOS 3D ESTEREOSCÓPICO



# Fundamentos – O que é 3D estereoscópico?

- Disparidade binocular
  - Duas perspectivas diferentes
  - Enxergamos somente uma imagem, com percepção de profundidade → **estereopsia**
- Duas imagens → **par estéreo**
  - Uma imagem para o olho esquerdo e outra para o olho direito
  - Deslocadas horizontalmente



# Fundamentos – O que é 3D estereoscópico?

- Equipamento especial para captura
  - Duas lentes, simulando a visão humana
- Reprodução
  - Pode requerer projetores e telas especiais, dependendo da técnica de visualização utilizada



Câmera para captura 3D estereoscópica  
(PANASONIC)

# Fundamentos – Como enxergar?



Quadro de filme estereoscópico (Shrek 3D)

- **Separação** do par estéreo
  - Auxílio de óculos
    - Passivo: anaglífico, polarizador
    - Ativo: obturador
  - Sem óculos
    - Monitores Autoestereoscópicos



# Fundamentos – Técnica Anaglífica

- Método mais simples para visualização estereoscópica
- Fusão das duas imagens em apenas uma, através de retirada de componentes de cor
- Óculos especiais com lentes semelhantes → filtro

Imagem do lado direito

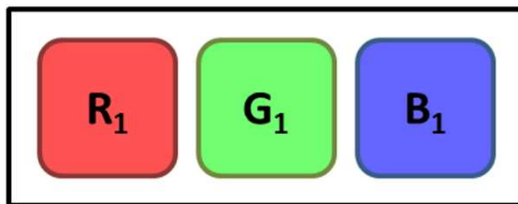
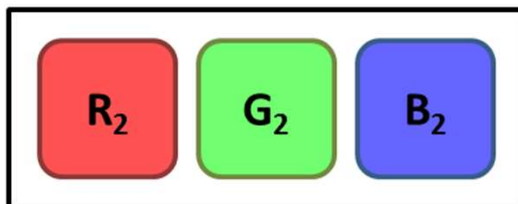
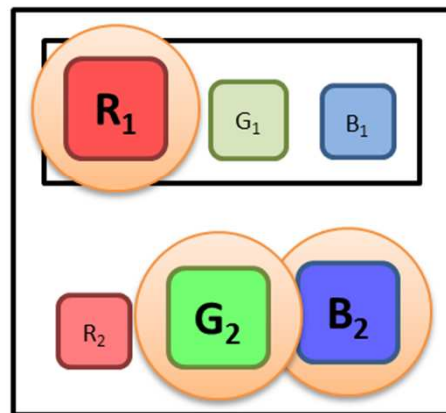


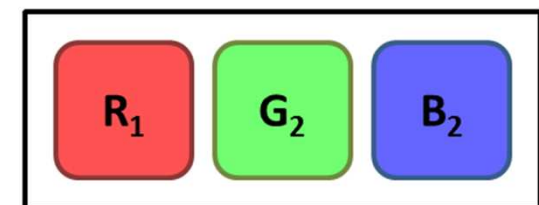
Imagem do lado esquerdo



Conversão anaglífica



Anáglifo vermelho-ciano



Processo de conversão anaglífica vermelho-ciano (Zingarelli, 2011 – adaptado)







# Fundamentos – Técnica Anaglífica

- Tipos:
  - Vermelho-ciano:  $R_1 G_2 B_2$
  - Verde-magenta:  $R_2 G_1 B_2$
  - Azul-amarelo:  $R_2 G_2 B_1$
  - Vermelho-azul:  $R_1 \_ B_2$
  - ....



# Fundamentos – Técnica Anaglífica

- Tipos:

- Vermelho-ciano:  $R_1G_2B_2$
- Verde-magenta:  $R_2G_1B_2$
- Azul-amarelo:  $R_2G_2B_1$
- Vermelho-azul:  $R_1\_B_2$
- ....

Mais comuns



# Fundamentos – Técnica Anaglífica

- Tipos:

- Vermelho-ciano:  $R_1 G_2 B_2$
- Verde-magenta:  $R_2 G_1 B_2$
- Azul-amarelo:  $R_2 G_2 B_1$
- Vermelho-azul:  $R_1 \_ B_2$
- ....

Melhores Resultados  
(Andrade & Goularte, 2009)





# Fundamentos – Técnica Anaglífica

- Tipos:

- Vermelho-ciano:  $R_1G_2B_2$
- Verde-magenta:  $R_2G_1B_2$
- Azul-amarelo:  $R_2G_2B_1$
- Vermelho-azul:  $R_1\_B_2$
- ....

Muito utilizado  
no começo.  
Péssima qualidade.



# Fundamentos – Links interessantes

- Comprar óculos
  - <http://www.3dshop.com.br/>
  - <http://www.tecnoglasses.com.br/>
- Youtube 3D
  - <http://www.youtube.com/3D>



# Fundamentos – Outros Métodos

- Luz Polarizada
  - Filtros polarizam o sinal de cada vídeo de modo diferente
  - Lentes dos óculos filtram o sinal de vídeo correspondente para cada olho
  - Tecnologia dos cinemas 3D
- Óculos Obturadores
  - Separação mecânica das imagens
  - Alternância das lentes entre transparente e opaca
    - Alta frequência
  - Equipamentos mais caros
- Monitores autoestereoscópicos
  - Película redireciona a luz em vários ângulos diferentes
    - Cada ângulo possui uma nova perspectiva da cena



Dúvidas até aqui?

**CHECKPOINT**



Ver para crer

**VÍDEO DEMO**

Projeto 1

# CRIAÇÃO DE IMAGEM ANÁGLIFA

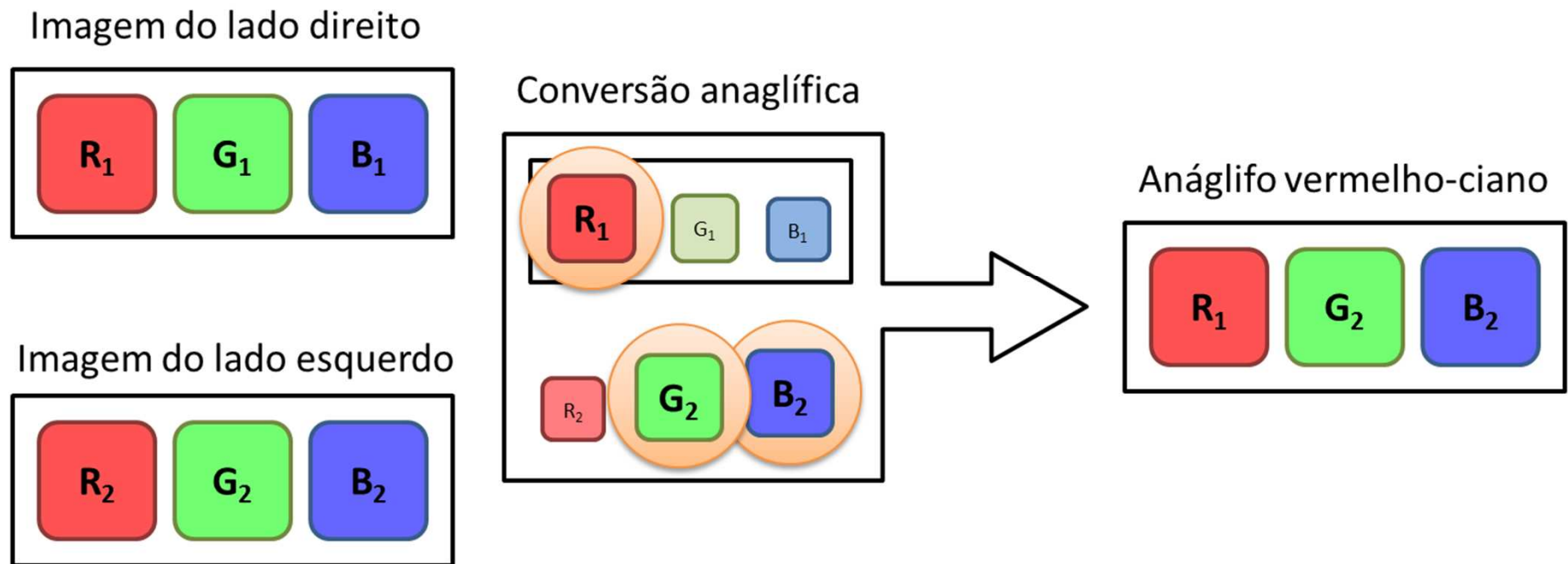


# Passos

- Abrir imagem (par estéreo)
- Criar nova imagem (anáglifo)
- Separar imagens
- Selecionar componentes de cor necessárias
  - Leitura das imagens
  - Criação de um anáglifo vermelho-ciano
- Gravar anáglifo



# Lembrando...



Processo de conversão anaglífica vermelho-ciano (Zingarelli, 2011 – adaptado)



Dúvidas até aqui?

**CHECKPOINT**





Mão na massa de novo

# VÍDEOS: FUNÇÕES BÁSICAS

# Operações

- Abrir vídeo
- Reproduzir Vídeo
- Gravar vídeo



# CvCapture

- Estrutura para se trabalhar com vídeos
  - Tanto do arquivo quanto de câmeras

```
CvCapture* cvCreateFileCapture(  
    const char* filename  
);
```

É necessário ter a DLL dos codecs para que o OpenCV saiba como abrir!



# CvQueryFrame()

- Pega o próximo frame do vídeo e armazena em memória em uma estrutura `IplImage`
- Não é necessário chamar `cvReleaseImage` para cada frame recebido

```
IplImage* cvQueryFrame(  
    CvCapture* capture  
);
```



# Limpeza

```
void cvReleaseCapture(  
    CvCapture** capture  
);
```



# Exercício 07

- Abrir vídeo e mostrar o primeiro frame do vídeo em uma janela



# Operações

- Abrir vídeo ✓
- Reproduzir Vídeo
- Gravar vídeo





Dúvidas até aqui?

**CHECKPOINT**



# Reprodução

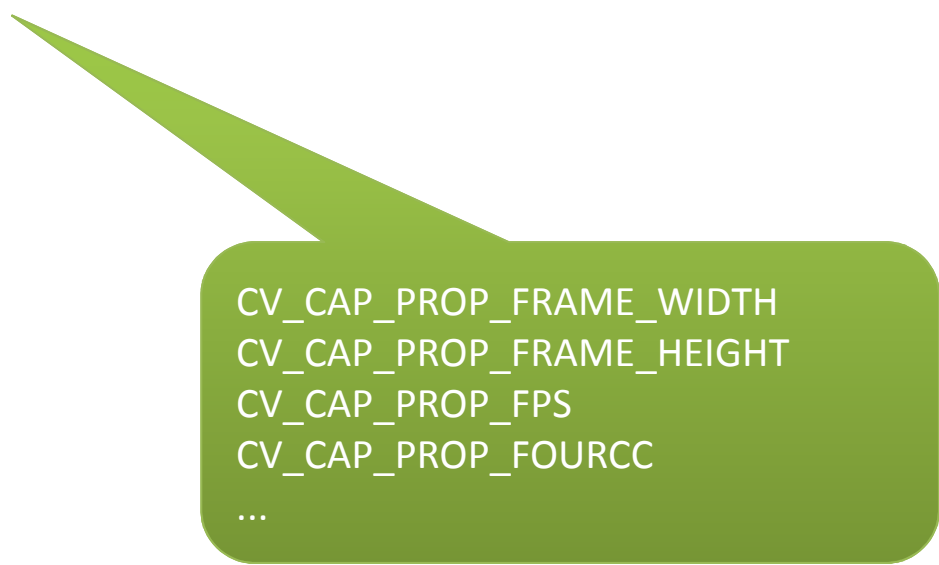
- Loop
  - Recupera um frame e o exhibe



# cvGetCaptureProperty()

- Propriedades do vídeo

```
double cvGetCaptureProperty(  
    CvCapture* capture,  
    int propId  
);
```



CV\_CAP\_PROP\_FRAME\_WIDTH  
CV\_CAP\_PROP\_FRAME\_HEIGHT  
CV\_CAP\_PROP\_FPS  
CV\_CAP\_PROP\_FOURCC  
...



# Exercício 08

- Fazer um player simples de vídeo



# Operações

- Abrir vídeo ✓
- Reproduzir Vídeo ✓
- Gravar vídeo



Dúvidas até aqui?

**CHECKPOINT**



# CvVideoWriter

- Estrutura para gravação

```
CvVideoWriter* cvCreateVideoWriter(  
    const char* filename,  
    int fourcc,  
    double fps,  
    CvSize frameSize,  
    int isColor=1  
);
```

Codec para  
gravar o  
vídeo

Vídeo colorido = 1  
Vídeo tons de cinza = 0

- Problema com o fourcc ☹
  - utilizar -1 e escolher o codec manualmente



# cvWriteFrame()

```
int cvWriteFrame(  
    CvVideoWriter*  writer,  
    const IplImage* image  
);
```





# cvReleaseVideoWriter()

```
void cvReleaseVideoWriter(  
    CvVideoWriter** writer  
);
```



# Exercício 09

- Abrir vídeo
- Aplicar uma conversão em cada frame
- Gravar frames em um novo vídeo



# Operações

- Abrir vídeo ✓
- Reproduzir Vídeo ✓
- Gravar vídeo ✓



Dúvidas até aqui?

**CHECKPOINT**





Projeto 2

# CRIAÇÃO DE VÍDEO ANÁGLIFO

# Passos

- Abrir vídeo
- Criar imagem (anáglifo)
- Para cada frame
  - Separar imagens
  - Selecionar componentes de cor necessárias
    - Leitura das imagens
    - Criação de um anáglifo vermelho-ciano
  - Gravar anáglifo
- <http://200.136.217.194/videoestereo/>



# Incrementando o projeto

- Par estéreo → 2 formatos
  - lado-a-lado (*side-by-side*)
  - acima-abaixo (*above-below*)
- Criar tratamento para vídeos acima-abaixo
- Adicionar novo parâmetro ao programa para que o usuário informe o tipo do vídeo



Última chance!

**DÚVIDAS?**





# Referências

- Andrade, L. A., Goularte, R. 2009. Anaglyphic stereoscopic perception on lossy compressed digital videos. *In Proceedings of the XV Brazilian Symposium on Multimedia and the Web (WebMedia '09)*. Fortaleza, v.1, n.1, 226-233. DOI=[10.1145/1858477.1858506](https://doi.org/10.1145/1858477.1858506)
- Bradski, G; Kaehler, A. – Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly, 2008.
- OpenCV Wiki - <http://opencv.willowgarage.com/wiki/>
- OpenNet Bulletin – China's Green Dam: The Implications of Government Control Encroaching on the Home PC. Disponível em [http://opennet.net/sites/opennet.net/files/GreenDam\\_bulletin.pdf](http://opennet.net/sites/opennet.net/files/GreenDam_bulletin.pdf) (Acesso em 16/09/2011)
- ZINGARELLI, M. R. U. – Reversão de imagens e vídeos estereoscópicos anaglíficos ao par estéreo original. 2011. 59f. Monografia de qualificação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2011.



# Iconografia

- Panasonic - [http://panasonic.biz/sav/broch\\_bdf/AG-3DA1\\_e.pdf](http://panasonic.biz/sav/broch_bdf/AG-3DA1_e.pdf)
- Shrek 3D - <http://3dindia.com/wp-content/gallery/shrek-3d-screen-shots/vlcsnap-2010-01-21-10h56m01s188.png>
- Acesso em 16/09/2011

# Links

- OpenCV
  - Rastreamento
    - <http://www.youtube.com/watch?v=RhPtylhWHFI>
  - Reconhecimento facial/gestos
    - <http://www.youtube.com/watch?v=B4dwu3si9x0>
  - Segmentação
    - <http://www.youtube.com/watch?v=ysSbYYWuAQg>
  - Kinect
    - <http://www.youtube.com/watch?v=0ITClcO8Wxg>



# Contato

- [zinga@icmc.usp.br](mailto:zinga@icmc.usp.br)
- Intermídia 6-209
- <http://www.icmc.usp.br/~zinga/>
- <http://viva3d.blogspot.com/>



OBRIGADO!

