

Processamento Digital de Imagens

Prof. Bogdan Tomoyuki Nassu



Nesta aula...

- Introdução.
- Informações sobre a disciplina.
- Algumas coisas básicas.

Professor: Bogdan T. Nassu

- Formação: ciência da computação.
 - Graduação e mestrado: UFPR (2002 e 2005).
 - Doutorado: Universidade de Tokyo (2008).
- Profissional:
 - UTFPR (2012~).
 - Professor adjunto, DAINF.
 - UFPR (2011~2012).
 - Pós-doutorado (CNPq).
 - Railway Technical Research Institute, Tokyo (2008~2011).
 - Pesquisador.
 - LACTEC (2001~2005).
 - Estagiário, bolsista (assistente de pesquisa).

www.dainf.ct.utfpr.edu.br/~nassu

Processamento digital de imagens

- Processamento?
- Digital?
- Imagens?



Áreas relacionadas

- Mundo físico.
 - Hardware.
 - Sensores.
 - Ótica.
- Processamento digital de sinais.
- Visão computacional.
 - (Qual a diferença?)
- Computação gráfica?!

Áreas relacionadas

- Mundo físico.
 - Hardware.
 - Sensores.
 - Ótica.
- Processamento digital de sinais.
- Visão computacional.
 - Não existe uma fronteira bem definida.
 - Para fins práticos, vamos deixar para visão computacional:
 - Análise de vídeos.
 - Descritores de regiões.
 - Aprendizado de máquina.
- Computação gráfica?!

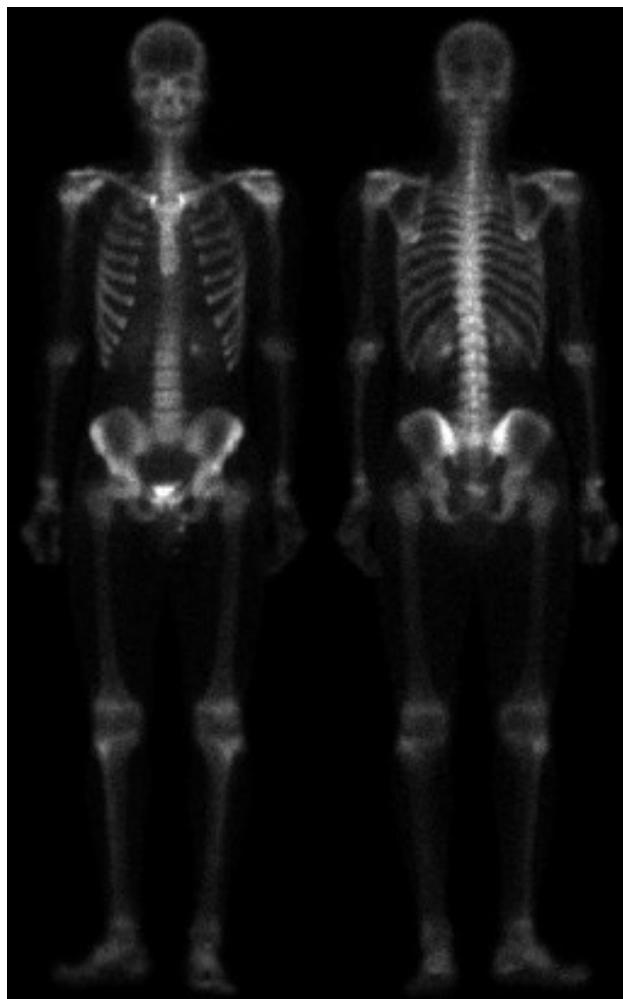
Exemplo de aplicação

- Aparelhos eletrônicos.



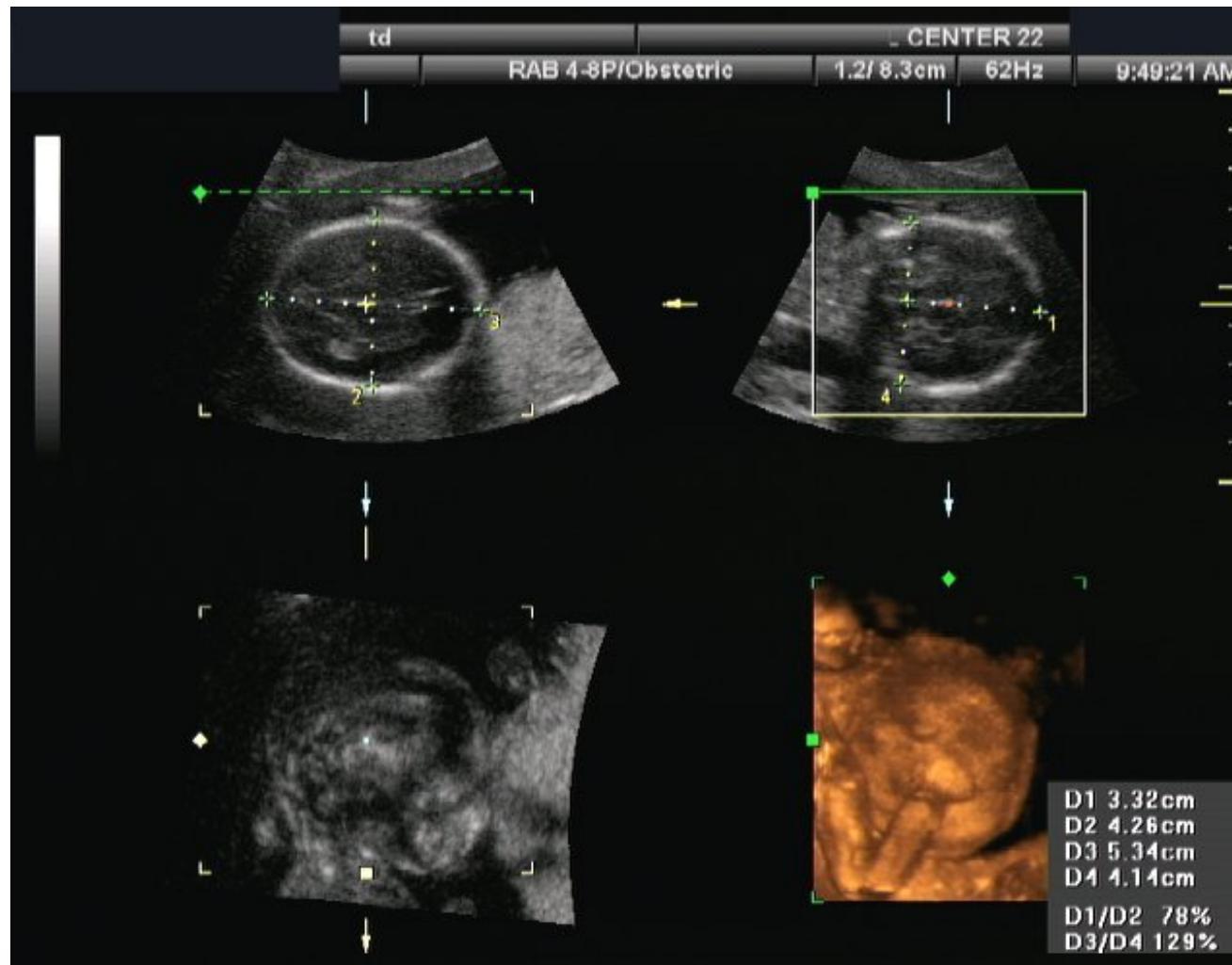
Exemplo de aplicação

- Visualização de imagens médicas.



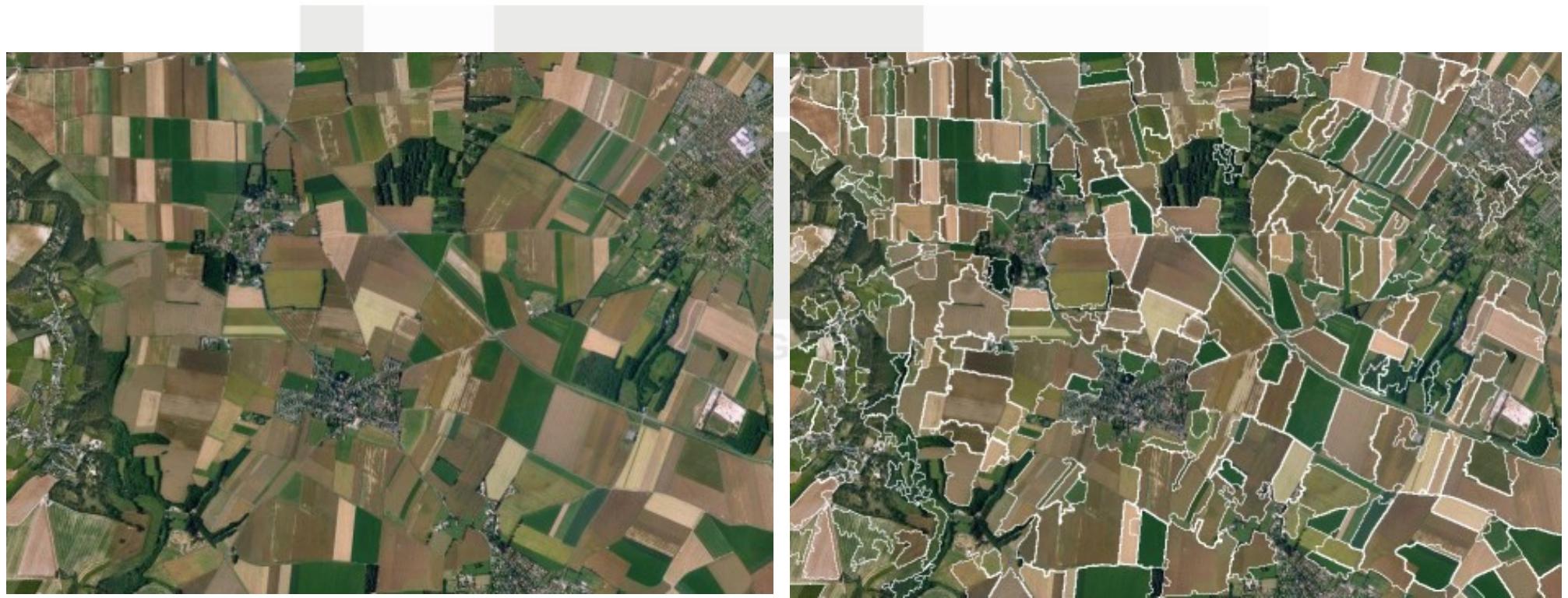
Exemplo de aplicação

- Visualização de imagens médicas.



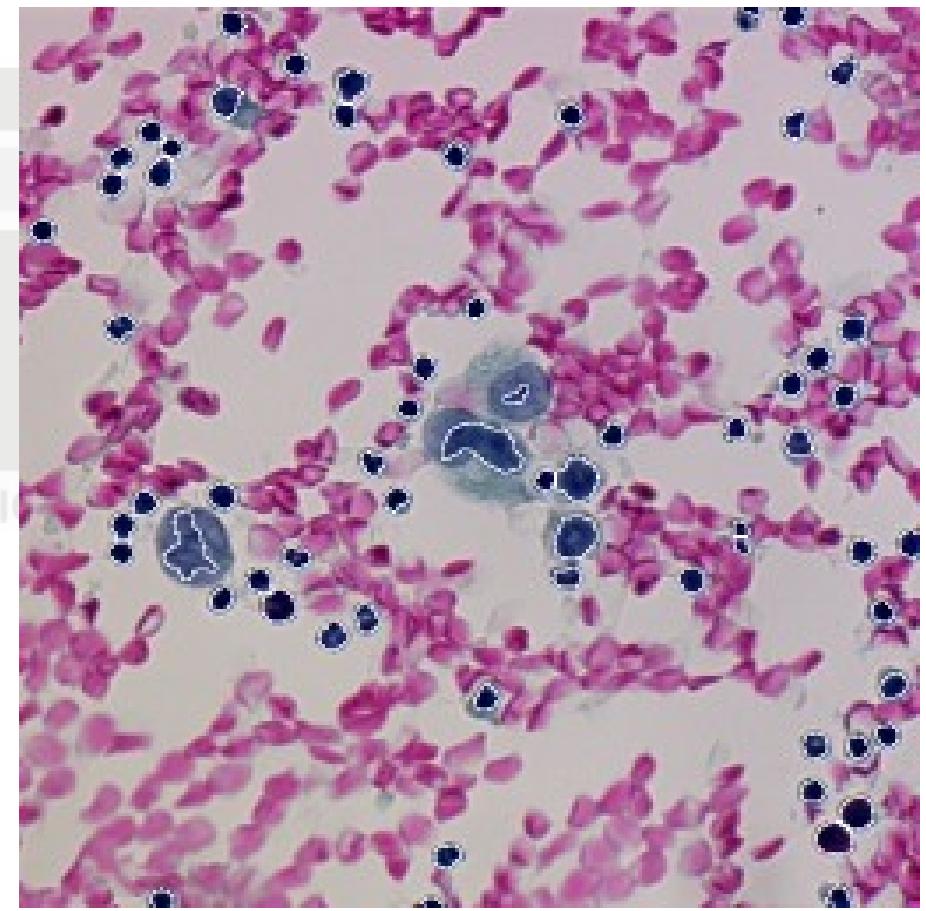
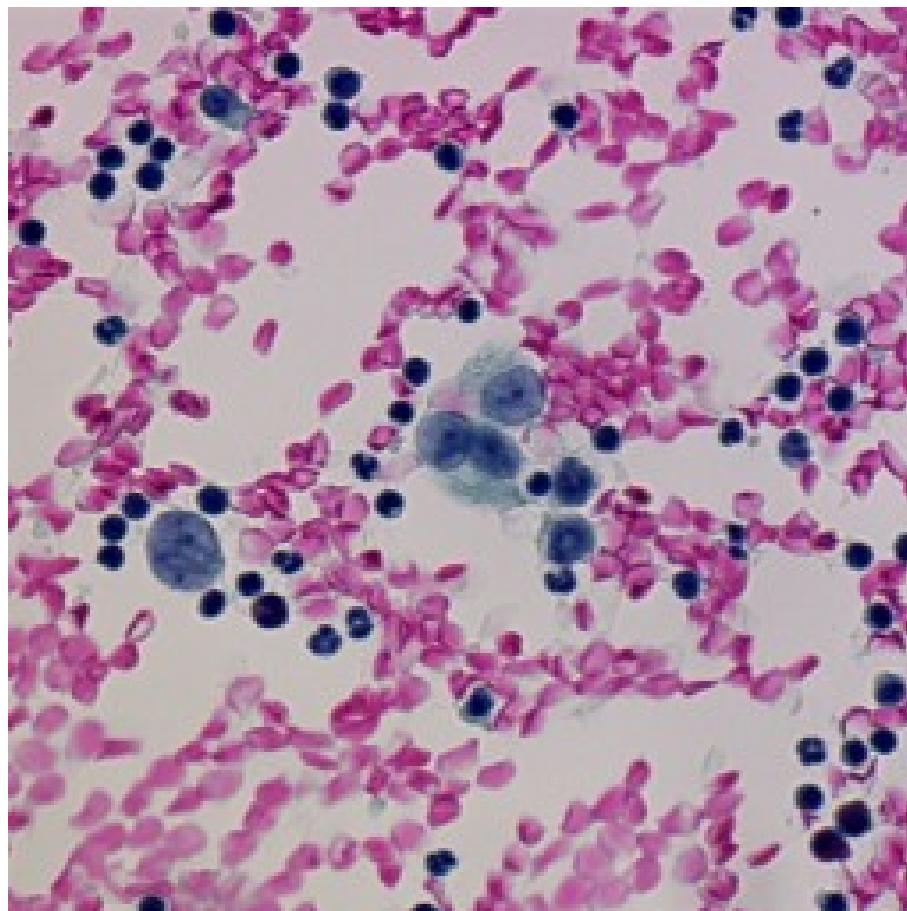
Exemplo de aplicação

- Geoprocessamento.



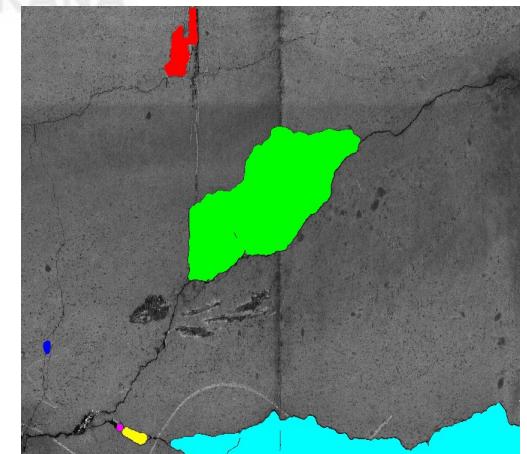
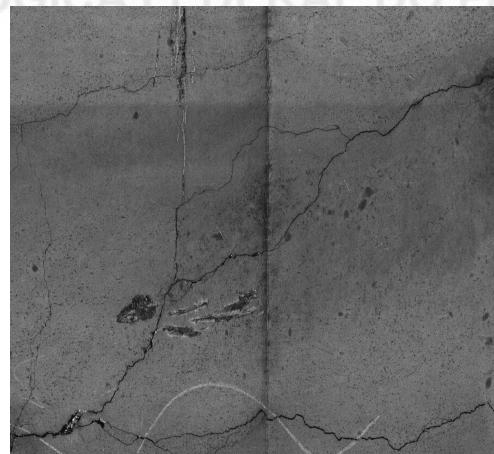
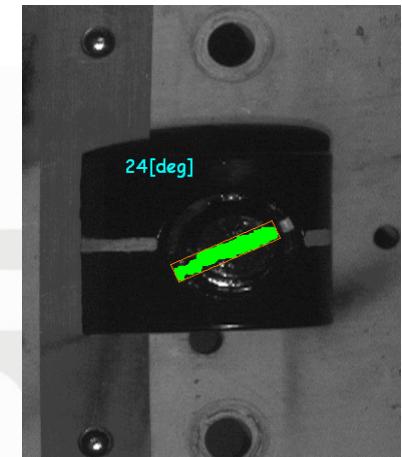
Exemplo de aplicação

- Visualização de imagens para experimentos científicos.



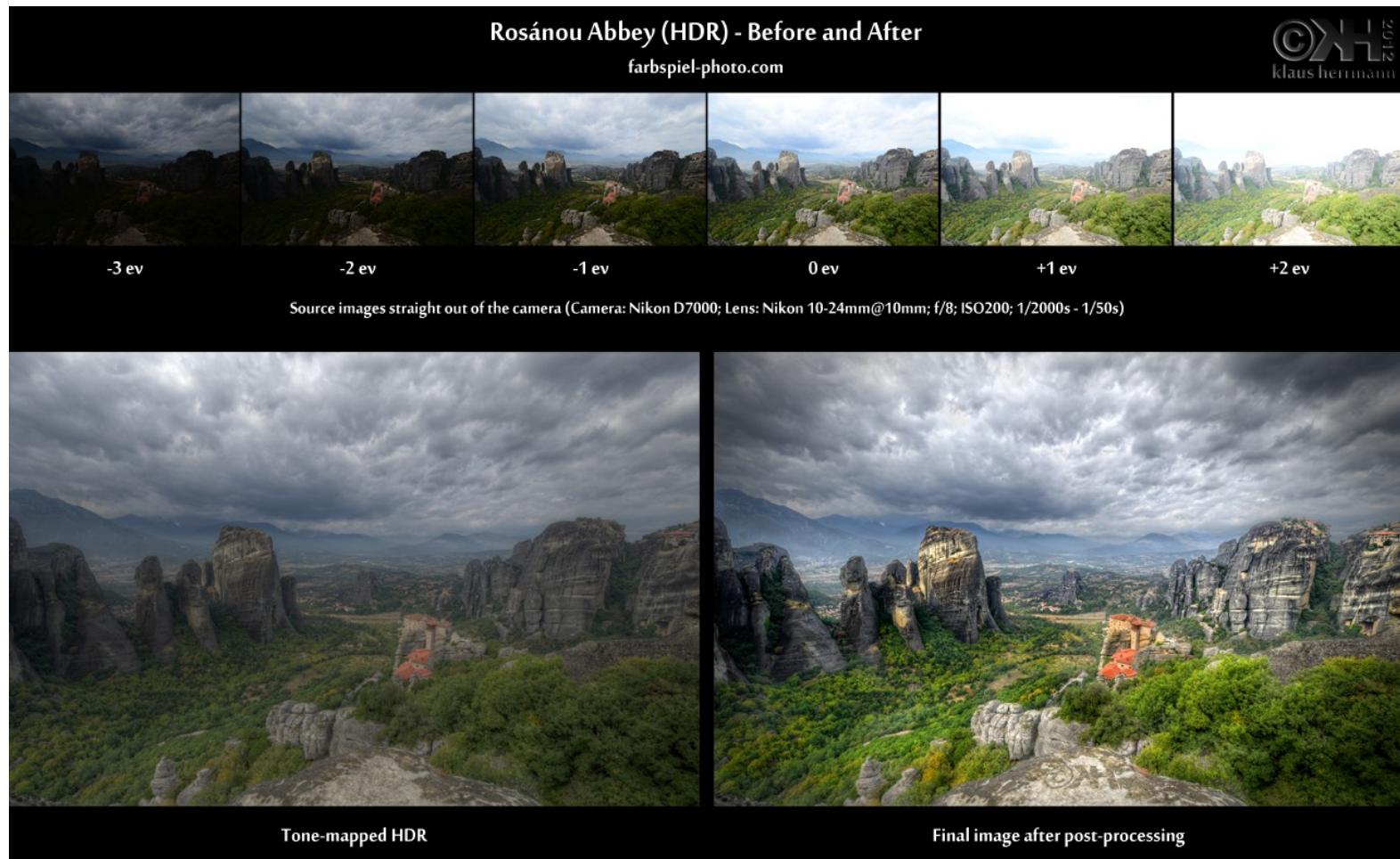
Exemplo de aplicação

- Inspeção de estradas de ferro.



Exemplo de aplicação

- *High dynamic range.*



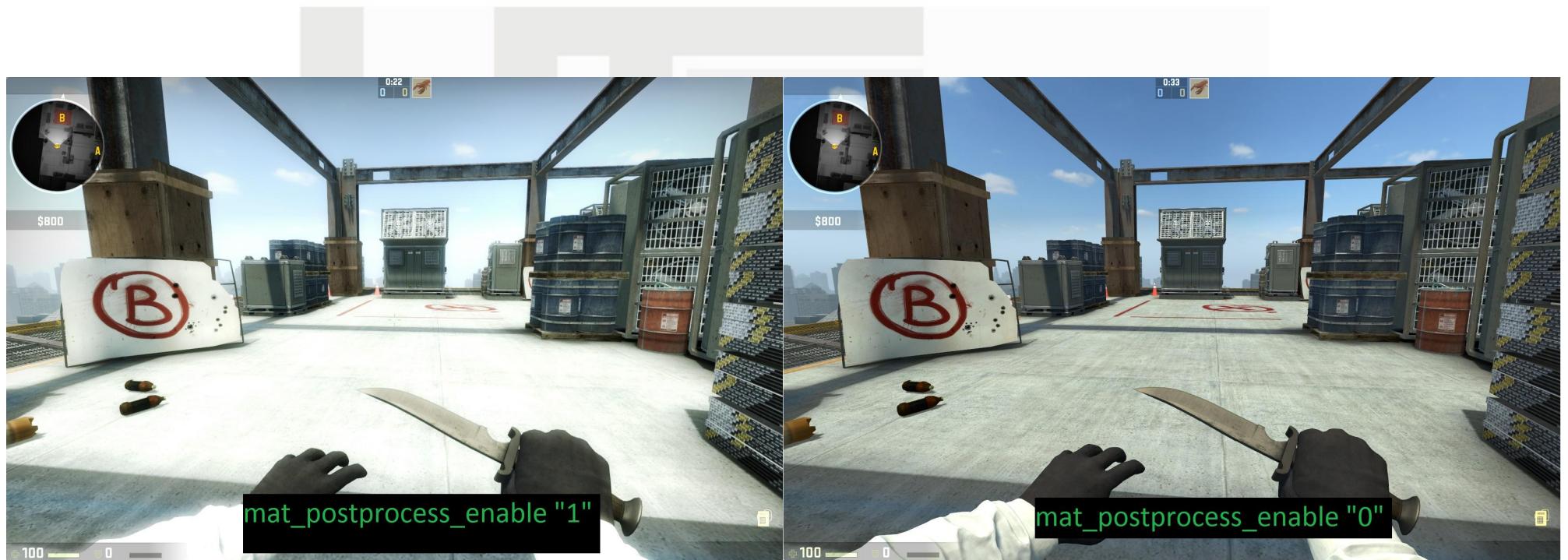
Exemplo de aplicação

- Efeito *depth of field* para imagens geradas por CG.



Exemplo de aplicação

- Efeito *bloom lighting* para imagens geradas por CG.



Exemplo de aplicação

- *Anti aliasing para imagens geradas por CG.*



Exemplo (ruim) de aplicação

- Er...



... e mais

- Técnicas de processamento de imagens servem como base para a área de visão computacional.



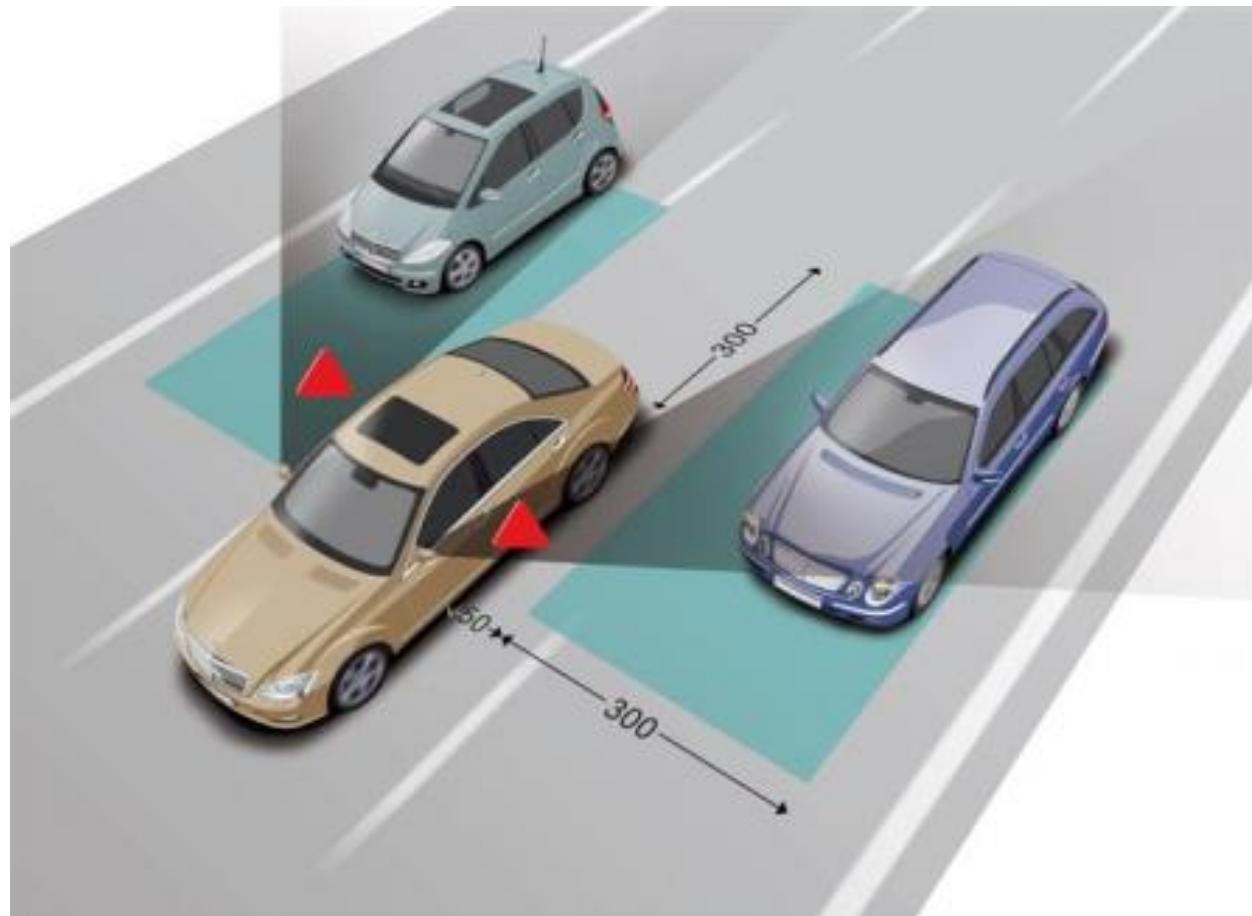
... e mais

- Técnicas de processamento de imagens servem como base para a área de visão computacional.



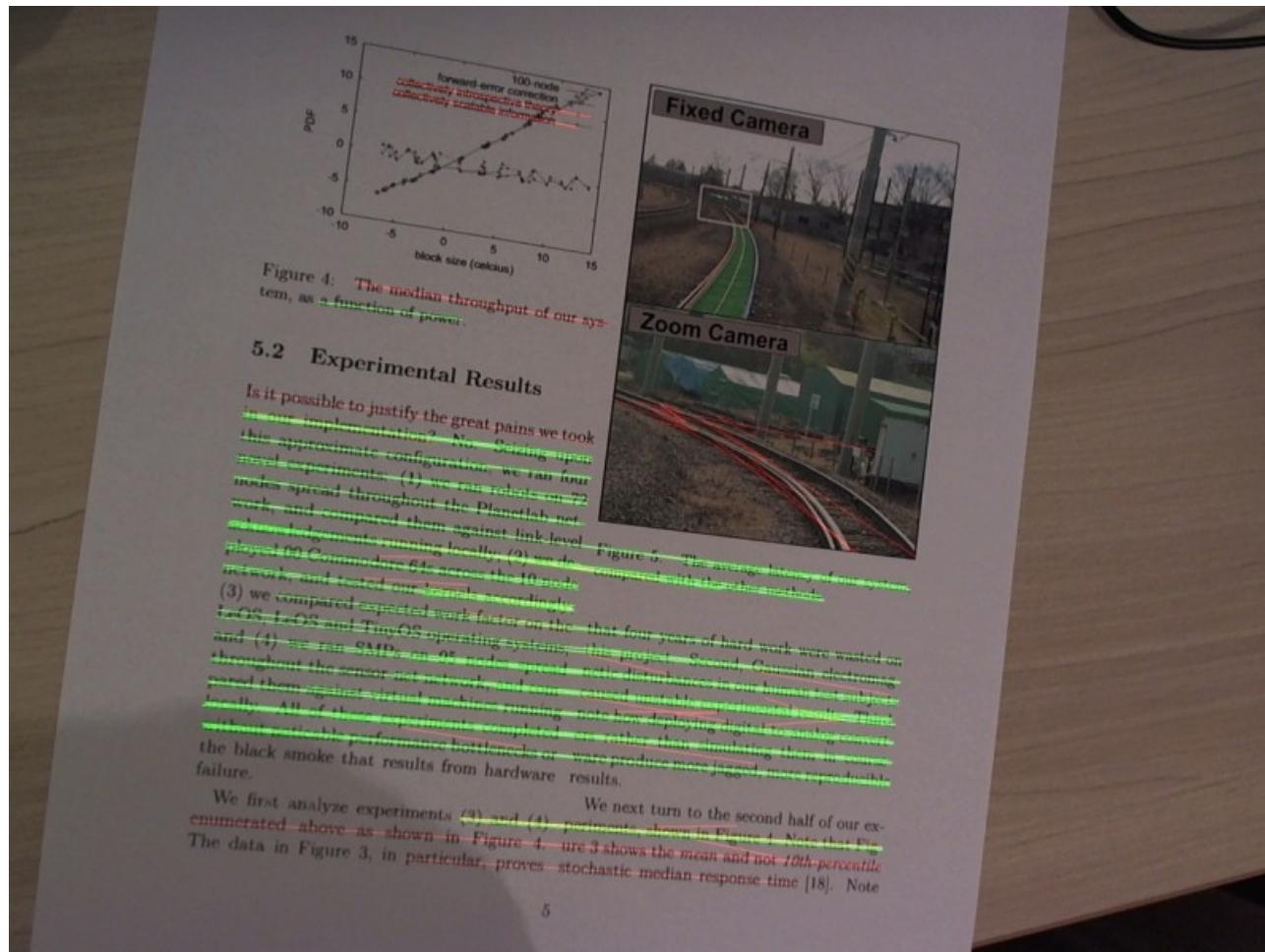
... e mais

- Técnicas de processamento de imagens servem como base para a área de visão computacional.



... e mais

- Técnicas de processamento de imagens servem como base para a área de visão computacional.



... e mais

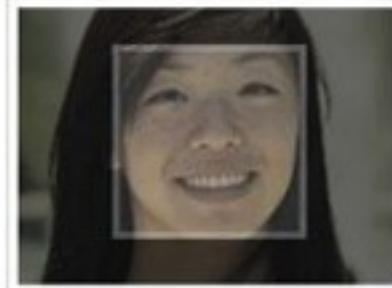
- Técnicas de processamento de imagens servem como base para a área de visão computacional.

Tag Your Friends

This will quickly label your photos and notify the friends you tag. Learn more



Who is this?



Who is this?



Who is this?



Who is this?



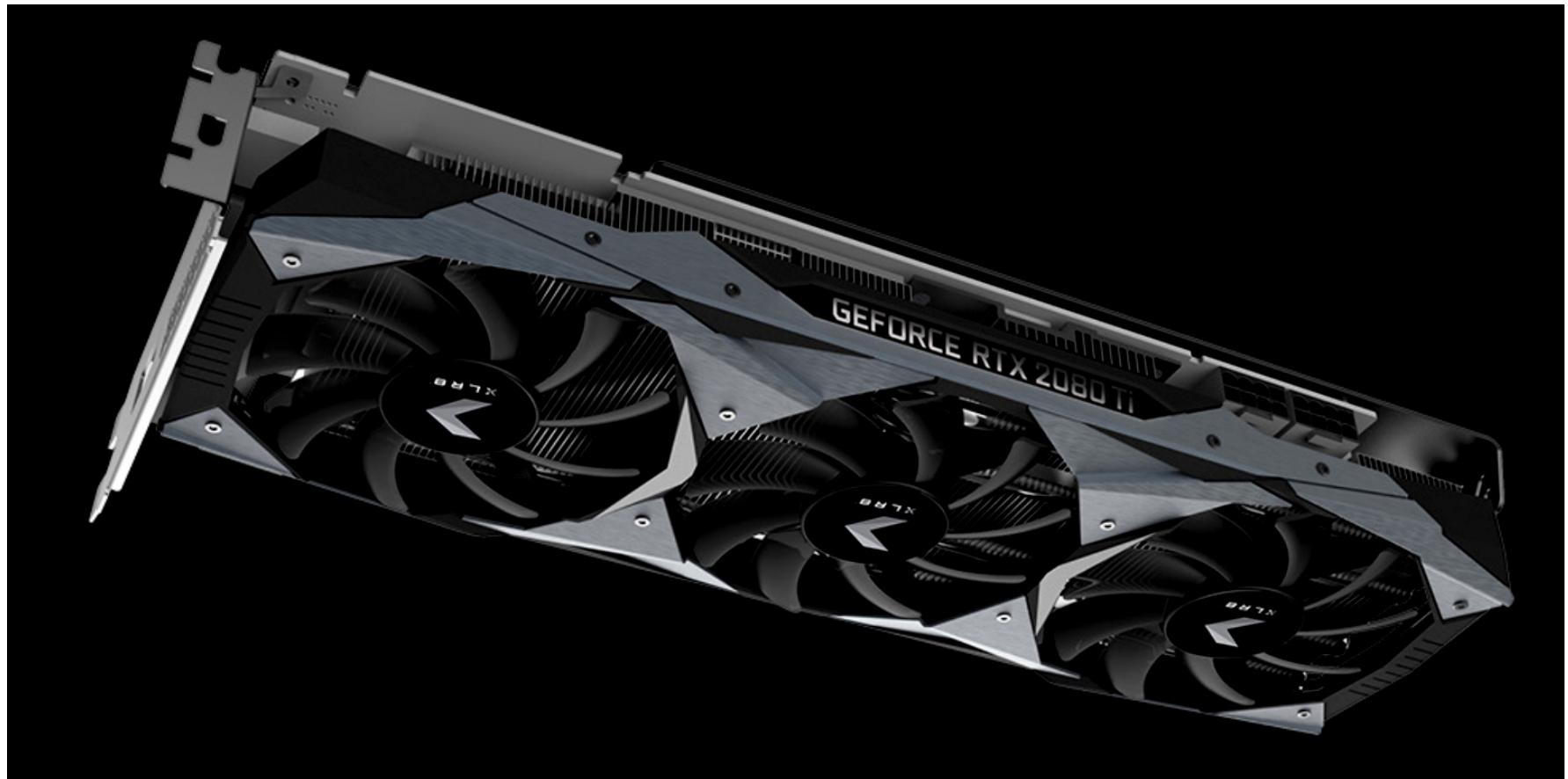
Who is this?



Who is this?

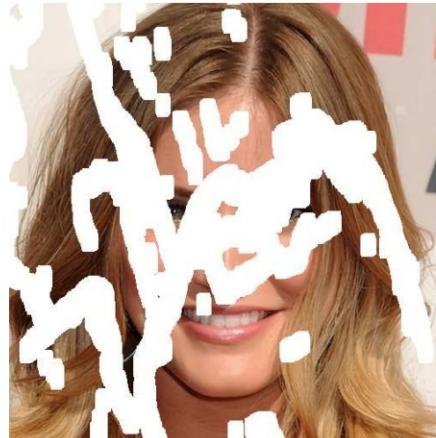
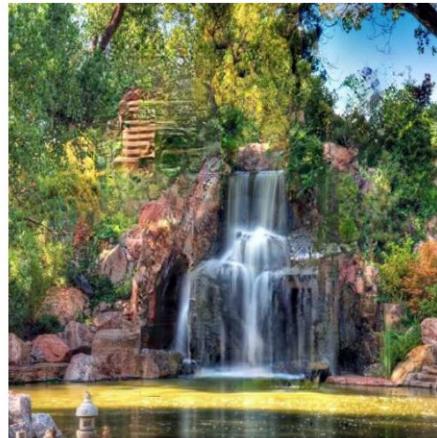
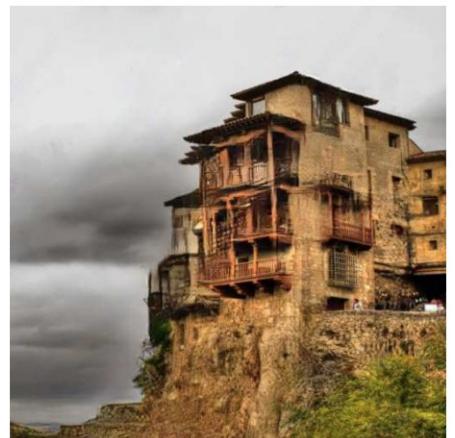
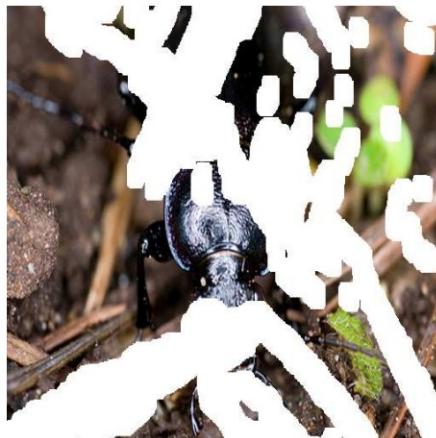
... e mais

- Aprendizado de máquinas + processamento de imagens = área quente.



... e mais

- Aprendizado de máquinas + processamento de imagens = área quente.



Disciplina

- Objetivo: *Capacitar o aluno para desenvolver sistemas que fazem uso de técnicas para o processamento digital de imagens, assim como identificar as classes de problemas que podem ser resolvidos com o uso de tais técnicas.*
- Abordagem:
 - Problema → técnicas.
 - Compreender os conceitos por trás de técnicas básicas.
 - Não vamos aprender a usar programas como Photoshop!!!
- Pré-requisitos: programação e estruturas de dados.
 - Um pouco de estatística e geometria analítica.
- Divisão do tempo:
 - Aulas “normais”: aproximadamente 1/2 do tempo.
 - Implementação de trabalhos: aproximadamente 1/3 do tempo.

Avaliação

- Trabalhos:

- 1 ou 2 pessoas.
- Apresentação para o professor.
- Pesos:
 - 3 implementações de algoritmos: 1.0 pontos cada.
 - 2 desafios: 1.6 pontos cada.
 - 1 projeto de tema aberto: 3.8 pontos.
- Linguagens aceitas:
 - C, C++, Python (recomendadas).
 - Java (por sua conta e risco).

- Nota final = nota dos trabalhos * multiplicador.

- Multiplicador: nota da prova + perguntas feitas em sala.
- Prova: intervalo [0, 0.9].
- Perguntas feitas em sala: intervalo [0, 0.5].
 - Porcentagem máxima de erros: 15%.

Bibliografia

- Não seguiremos o roteiro de um livro específico.
- Alguns livros interessantes:
 - Gonzalez e Woods, *Processamento Digital de Imagens*.
 - Shapiro e Stockman, *Computer Vision*.
 - Szeliski, *Computer Vision – Algorithms and Applications*.
 - Bradski e Laehler, *Learning OpenCV*.
- Artigos de congressos e periódicos da área.

Ferramentas

- Comunicados, entregas, etc.: Moodle do DAINF.

moodle.dainf.ct.utfpr.edu.br

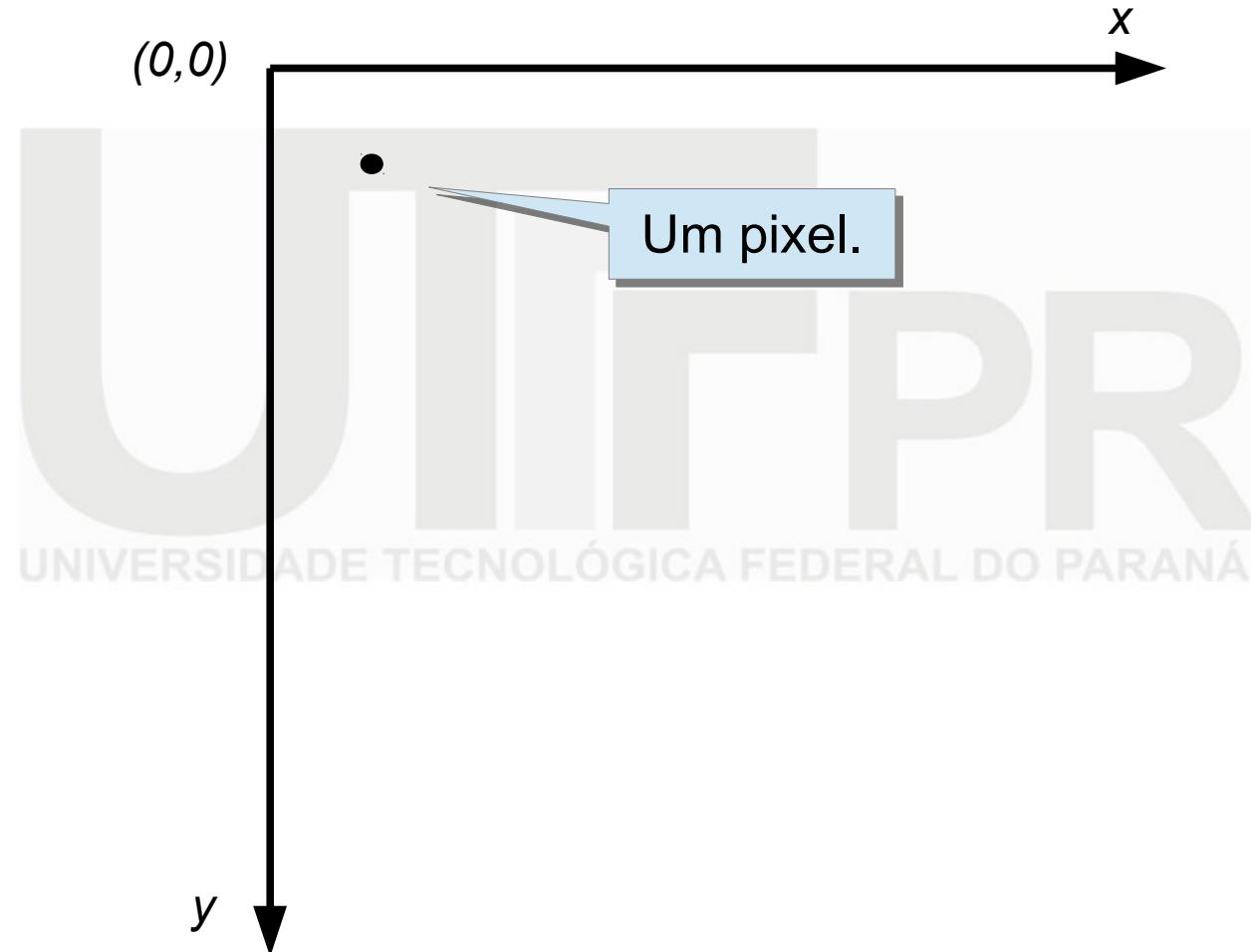
chave: nassu

- Não usaremos bibliotecas ou linguagens especializadas.
- O professor disponibilizará arquivos básicos em C para manipulação de imagens no formato bmp.
 - Implementações de outros algoritmos serão disponibilizadas durante o semestre.
- Quem quiser, pode usar a biblioteca OpenCV para o projeto final.
- (voltaremos a isso em breve).

Representando uma imagem: a representação clássica

- Uma imagem é uma função $f(x,y)$ que indica um valor para cada posição (x,y) no espaço.
- Cada ponto que forma a imagem é um *pixel*.
- O valor de $f(x,y)$ é a *intensidade* do pixel na posição (x,y) .
- O eixo y é invertido, comparado ao plano cartesiano tradicional.
- Imagens são *discretas*, i.e. todos os valores são inteiros ($f: \mathbb{Z}^2 \rightarrow \mathbb{Z}$).
 - Eventualmente, é útil usar coordenadas inteiras mas intensidades reais ($f: \mathbb{Z}^2 \rightarrow \mathbb{R}$), ou mesmo uma função contínua ($f: \mathbb{R}^2 \rightarrow \mathbb{R}$).
 - Por que?!
- Esta representação é uma das “heranças” da área de processamento de sinais para o processamento de imagens.

Representando uma imagem: a representação clássica

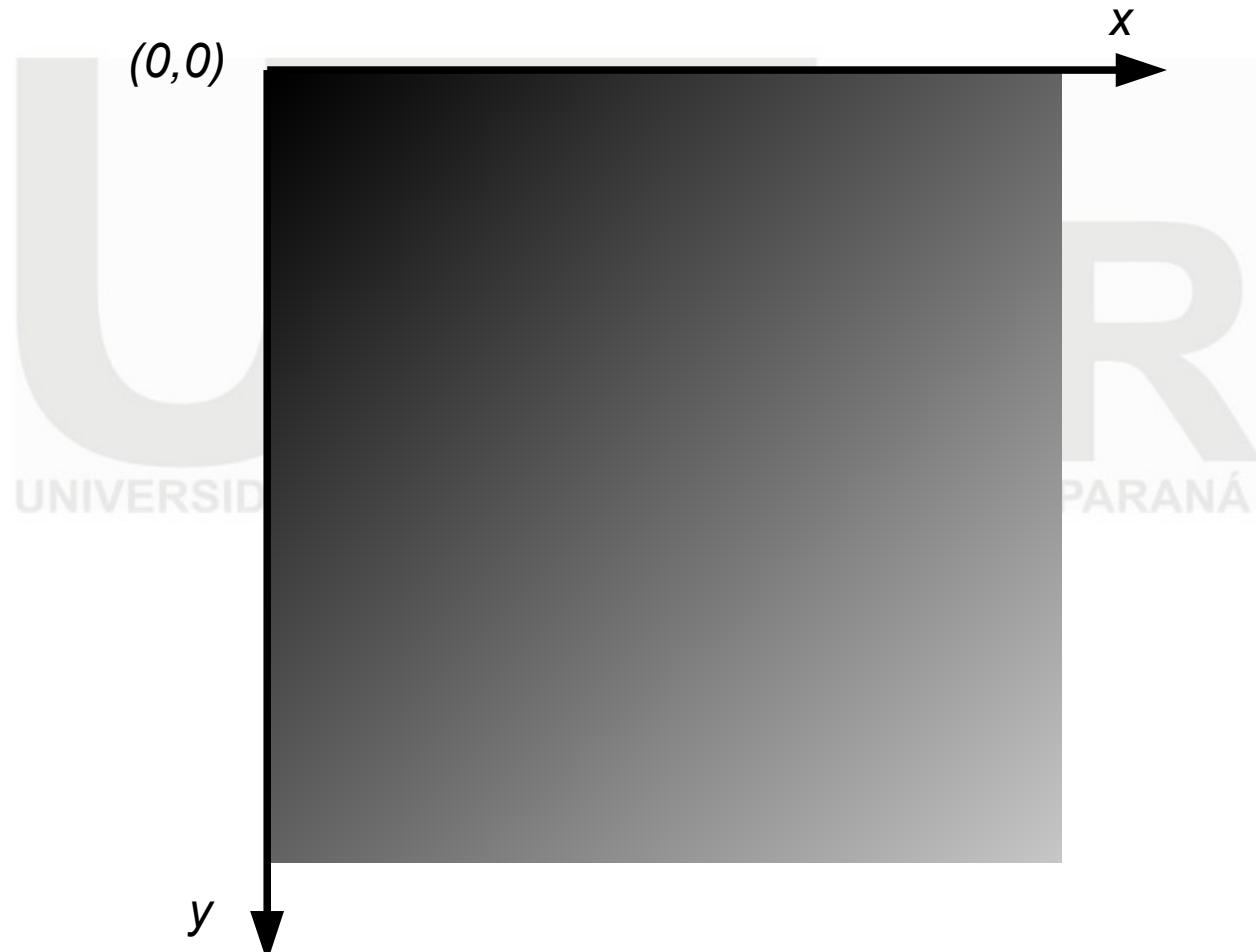


Representando uma imagem: a representação “computacional”

- Uma imagem é uma matriz $I[y][x]$, com cada posição da matriz indicando a intensidade do pixel na linha y e na coluna x .
 - Normalmente, 1 pixel = 1 byte.
 - 8 bits por pixel, ou 8bpp.
 - Valores no intervalo [0, 255].
 - Outras representações comuns:
 - 2 bytes com sinal.
 - floats no intervalo [0,1] (representação normalizada).
- A intensidade normalmente indica um tom de cinza.
 - Mais próximo de 0 → mais escuro.
 - Representação conhecida como *escala de cinza (grayscale)*.

Representando uma imagem: exemplo

- Uma imagem 100x100, de 8 bpp, onde $I[y][x] = y+x$, ou $f(x,y) = x+y$.

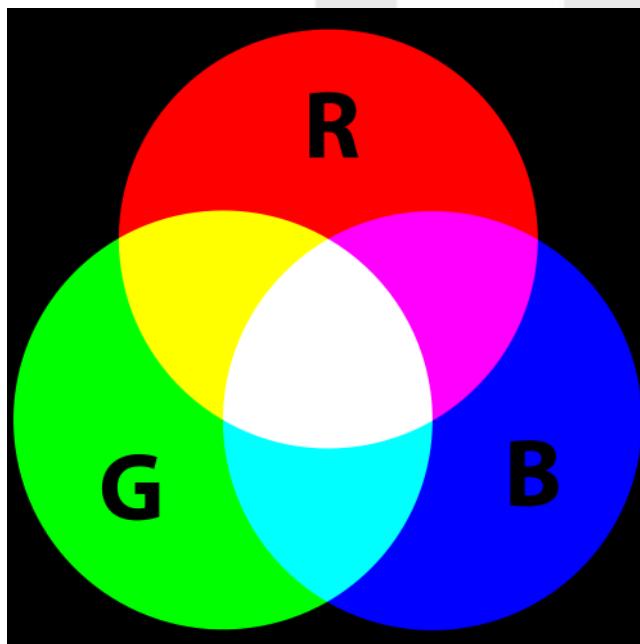


Usando cores

- Muitas técnicas clássicas de processamento de imagens consideram imagens em escala de cinza, por várias razões:
 - Simplicidade conceitual.
 - Processamento eficiente.
 - Limitações dos equipamentos mais antigos.
 - Características dos sensores.
 - Às vezes, o que se mede não é a intensidade da luz, mas outra medida com um único valor por pixel (ex.: temperatura, distância).
 - Muitas tarefas podem ser resolvidas ignorando as cores.
- Mas imagens coloridas são muito comuns!

O modelo RGB

- Normalmente, representamos uma cor como uma soma de 3 componentes (cores primárias): vermelho, verde e azul.
 - Modelo RGB.
 - É um modelo *aditivo*.
 - Base nos receptores do olho humano.



Tem emissores
para R, G e B.



Tem receptores
(sensores)
para R, G e B.

Representando uma imagem colorida

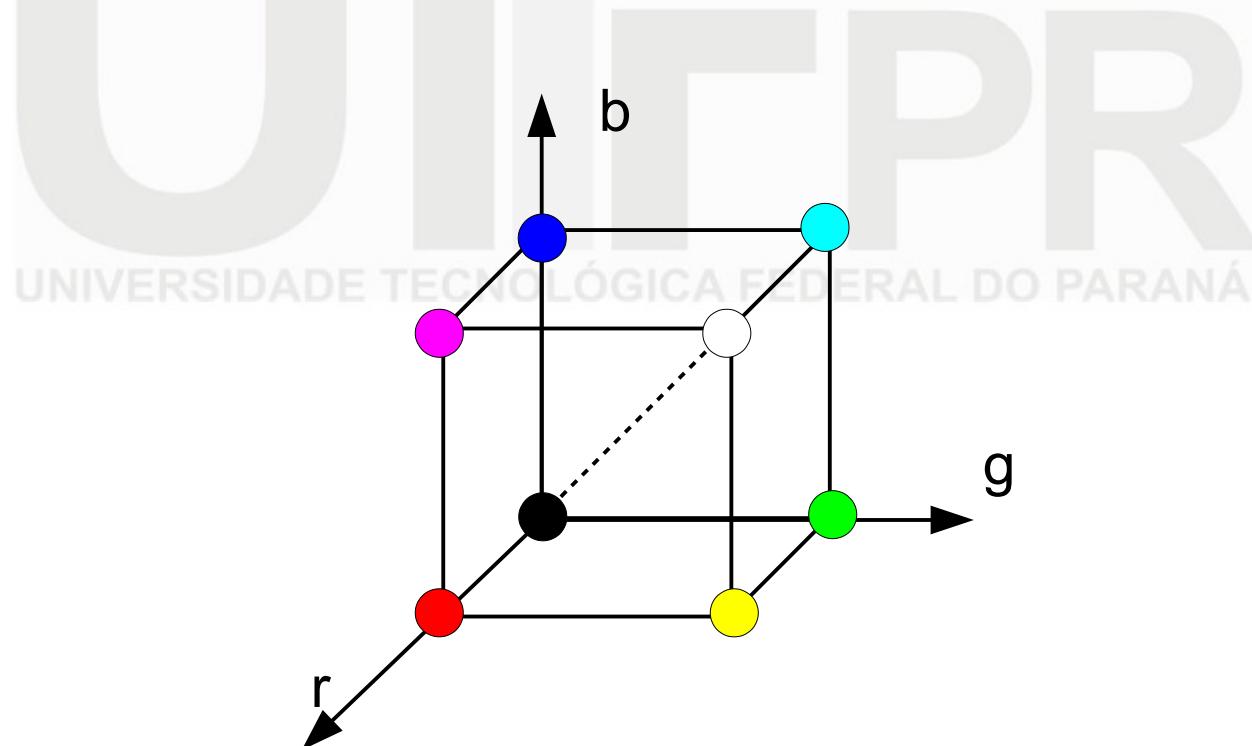
- Dizemos que uma imagem colorida tem 3 canais.
 - Cada pixel $f(x,y)$ é associado a uma tupla de 3 valores ($f: \mathbb{Z}^2 \rightarrow \mathbb{Z}^3$).
 - Na representação matricial, podemos acrescentar uma dimensão à imagem: $I[c][y][x]$, onde c é o canal.
- O modo mais usado tem 1 byte para cada canal (24bpp).
 - Quantas cores podemos representar com 24bpp?

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ



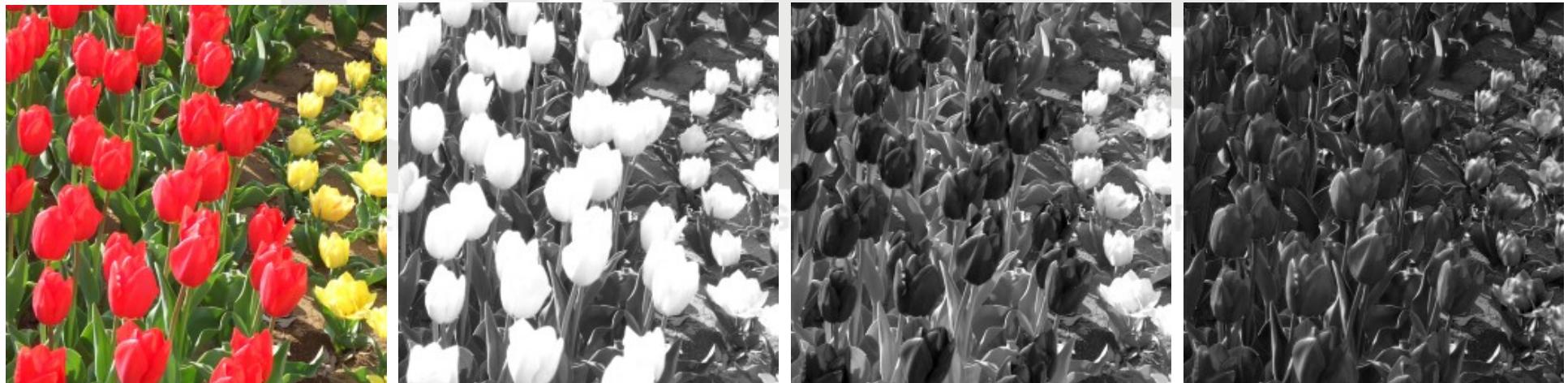
Visualizando o espaço RGB

- Em imagens coloridas, $f(x,y)=(r,g,b)$.
 - Cada cor é um ponto em um espaço vetorial.
 - O espaço RGB é um cubo.
 - Tons de cinza ficam na diagonal com origem em $(0,0,0)$ e final em (m,m,m) , onde m é o valor máximo para a intensidade.

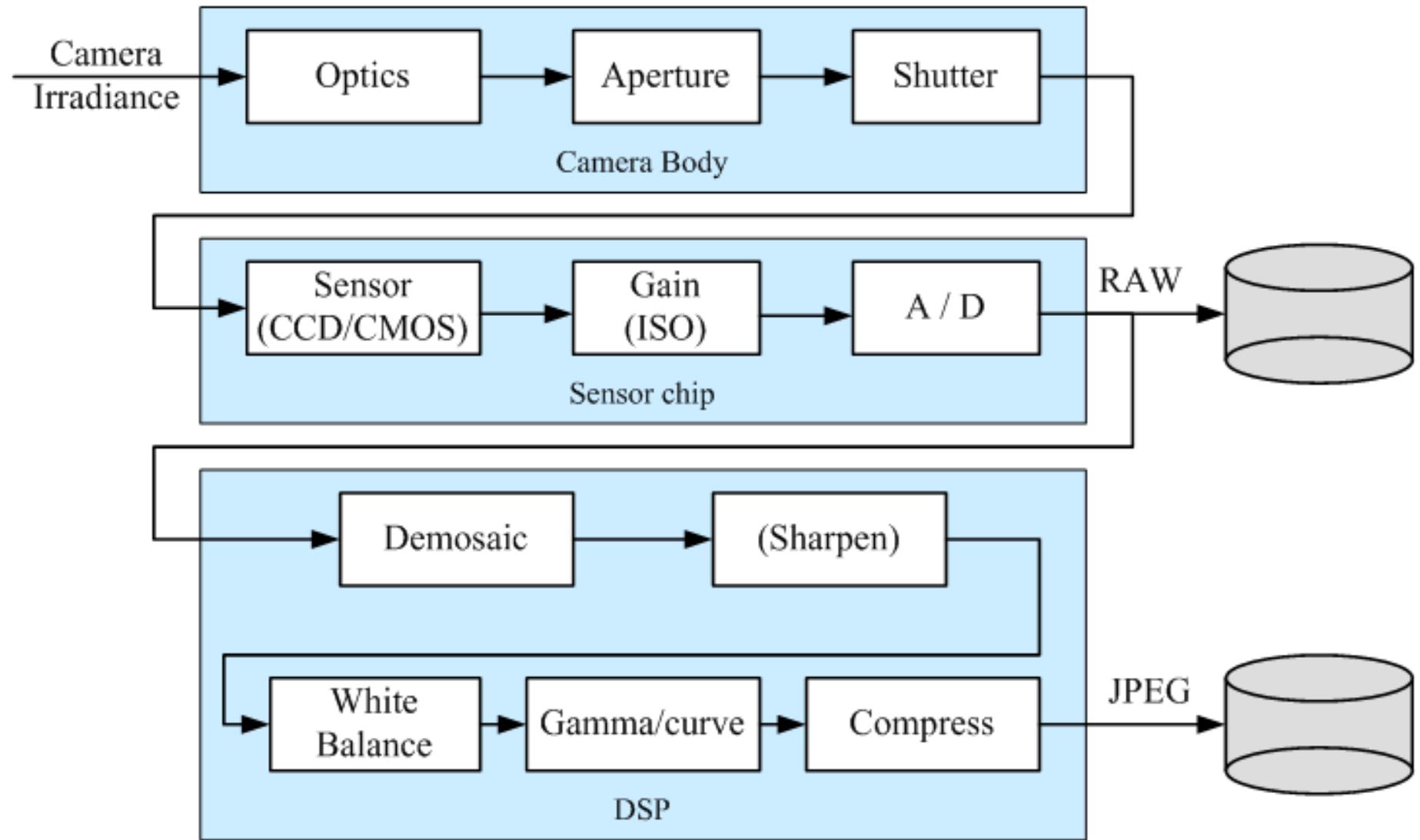


Separando os canais

- Quando separamos uma imagem em 3 canais, o resultado são 3 imagens de 1 canal, com a intensidade de um pixel para um canal indicando a quantidade daquele componente que está presente.



Como uma imagem é capturada?

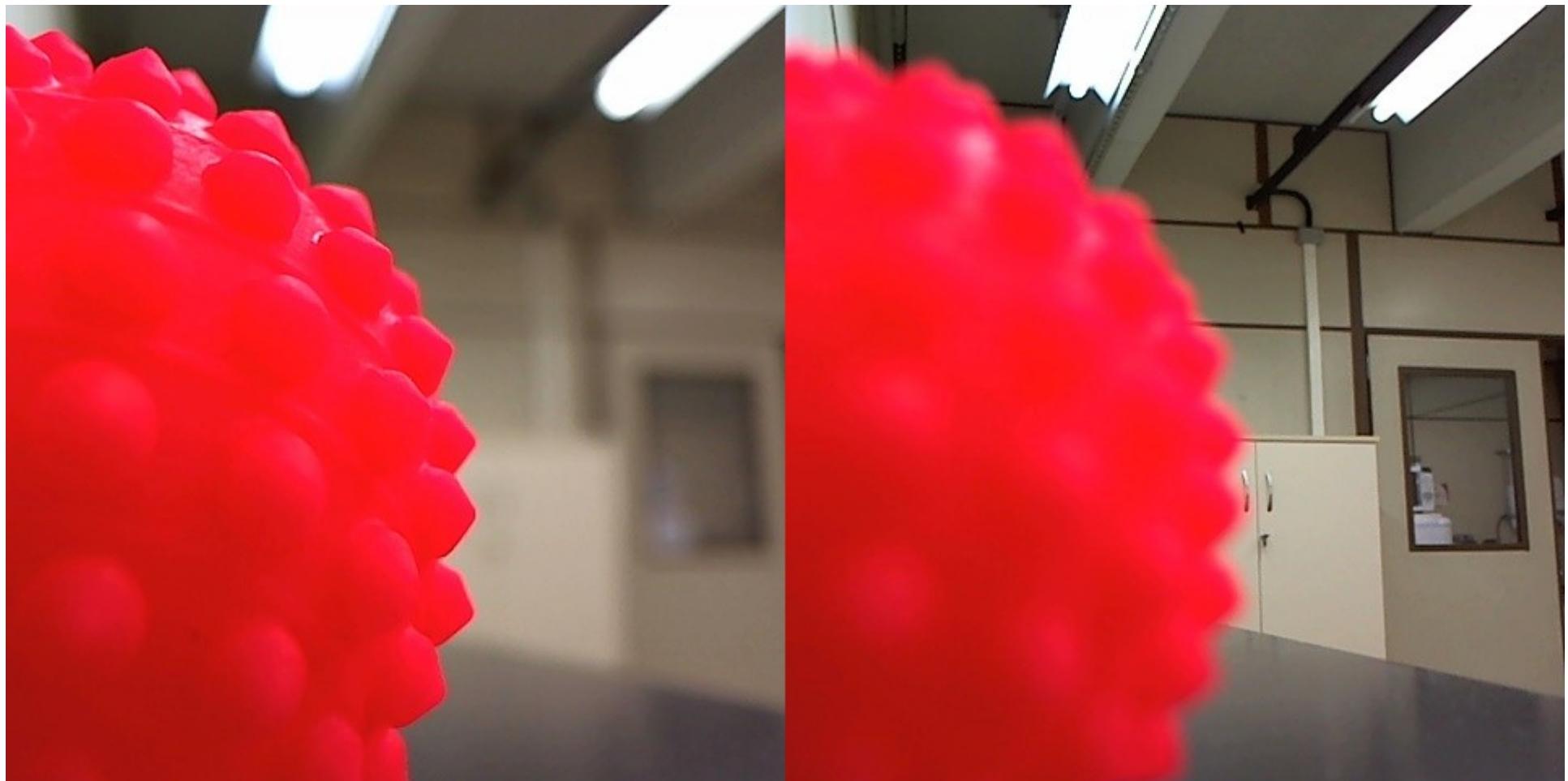


Como uma imagem é capturada?

- De maneira muito resumida...
 - A luz incide sobre uma superfície.
 - A superfície reflete a luz.
 - A luz refletida passa por uma lente e por uma pequena abertura na câmera, incidindo sobre sensores.
 - Os fótons são coletados pelos sensores por um determinado período de tempo (chamado de tempo de exposição).
 - Os dados dos sensores são convertidos para o domínio digital.
- As câmeras que usamos no dia a dia costumam automatizar certos controles, além de processar as imagens capturadas.
 - Autofoco.
 - Controle automático de balanceamento de cor (white balance).
 - Compressão.
- Existem vários tipos e organizações para os sensores.

A imagem é influenciada...

- ... pela distância focal.



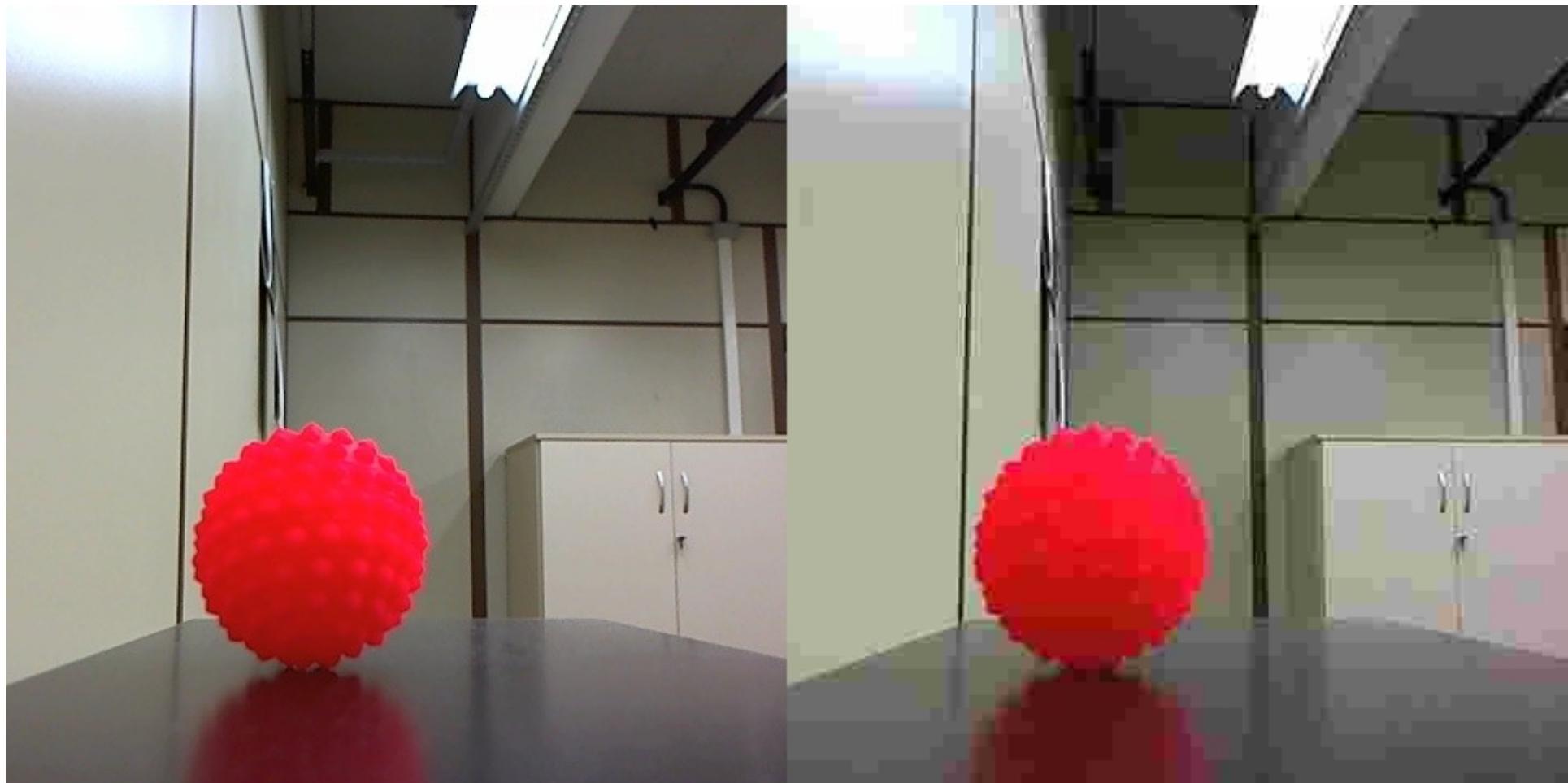
A imagem é influenciada...

- ... por distorções radiais da lente (tipo “almofada”, tipo “barril”).



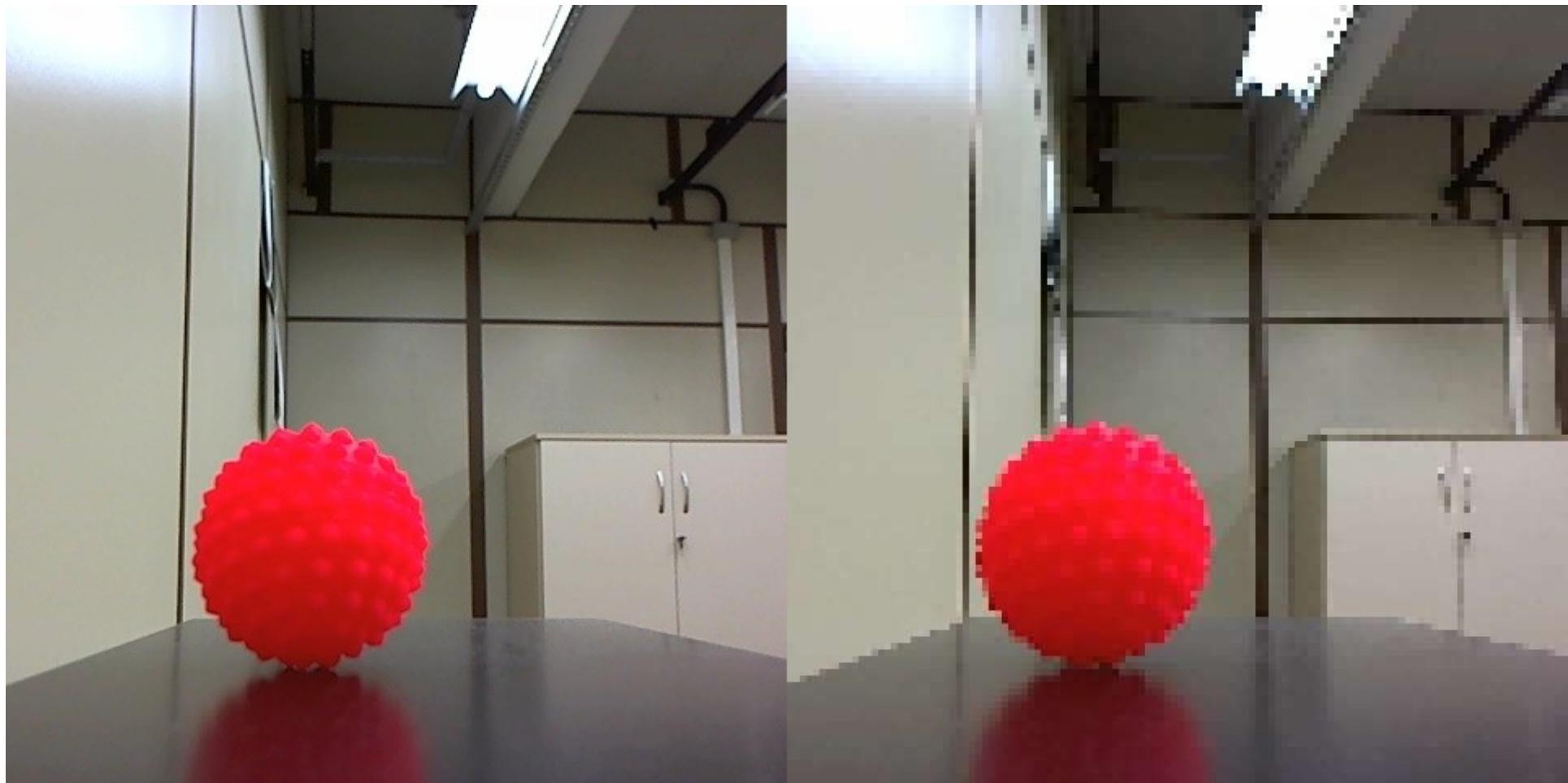
A imagem é influenciada...

- ... pela compressão.



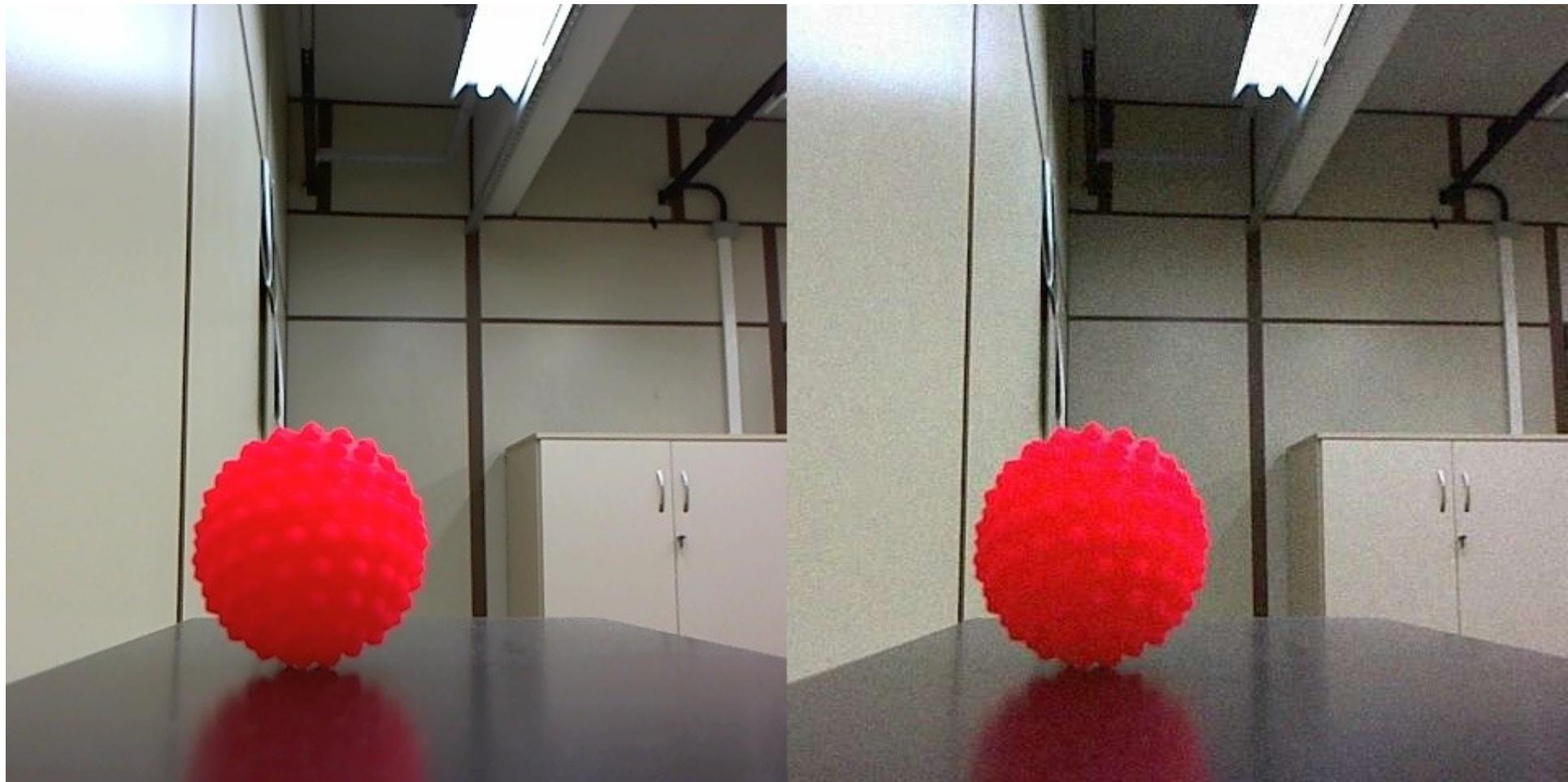
A imagem é influenciada...

- ... pela resolução.



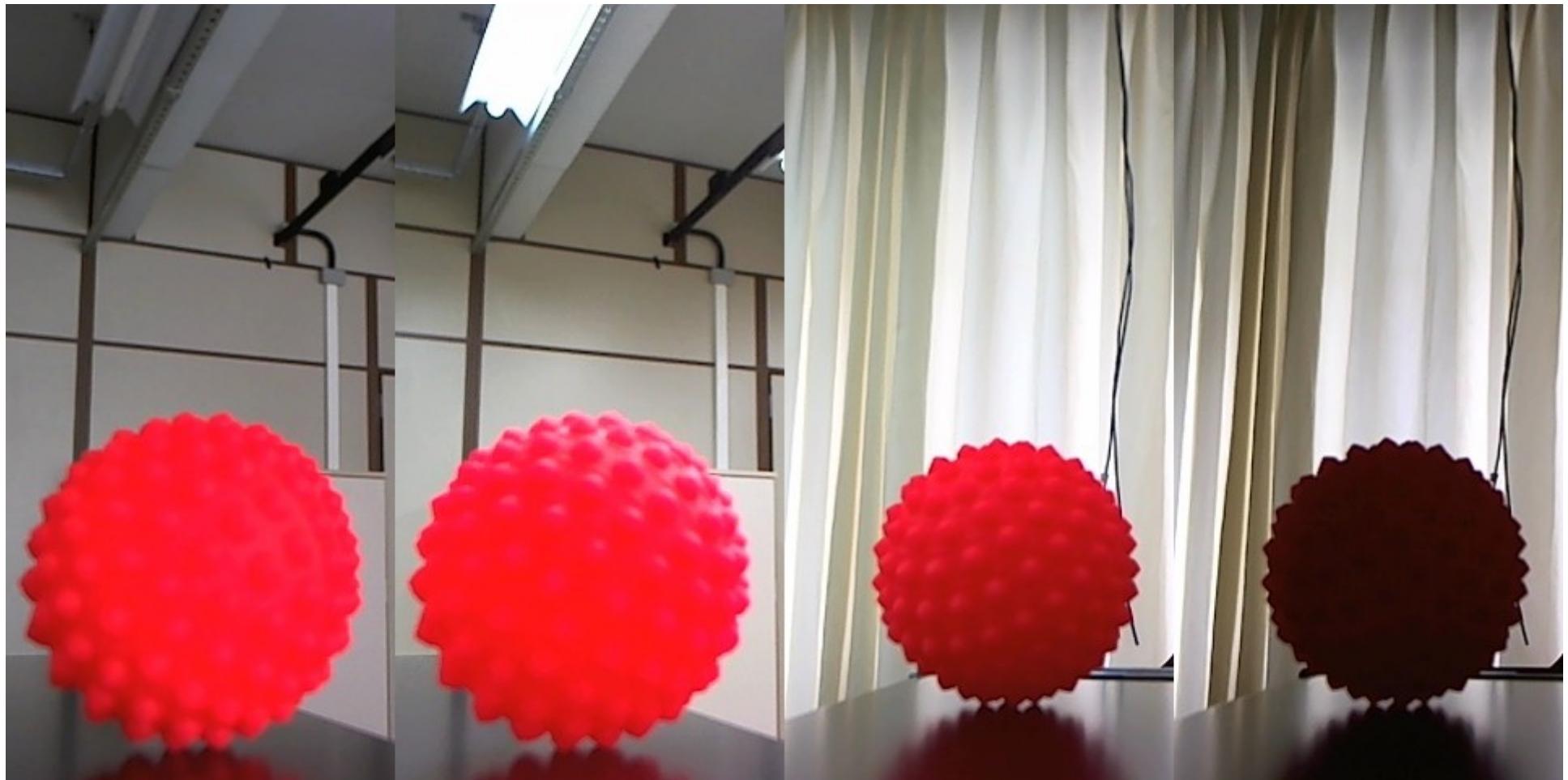
A imagem é influenciada...

- ... por ruídos.



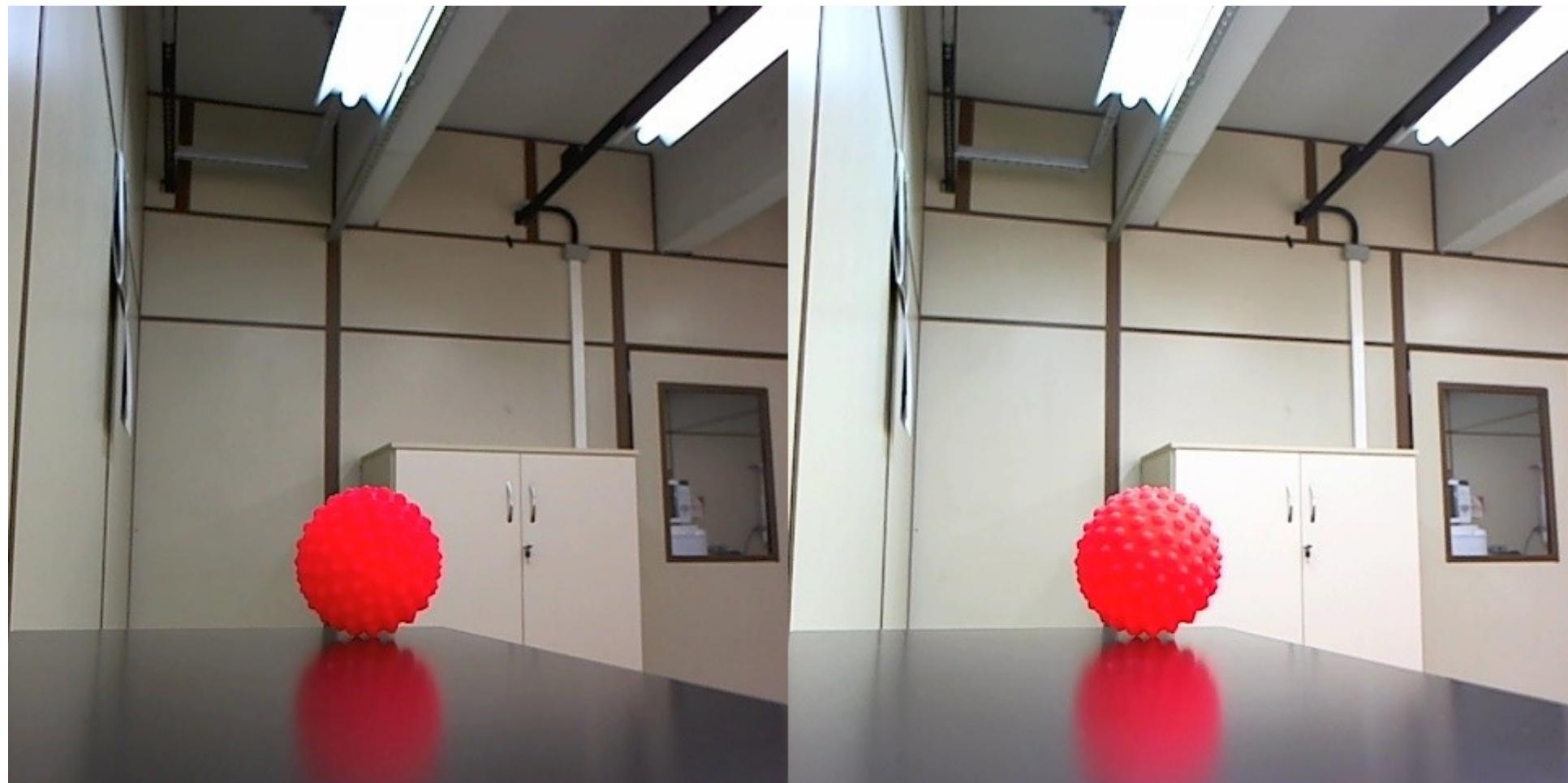
A imagem é influenciada...

- ... pela iluminação.



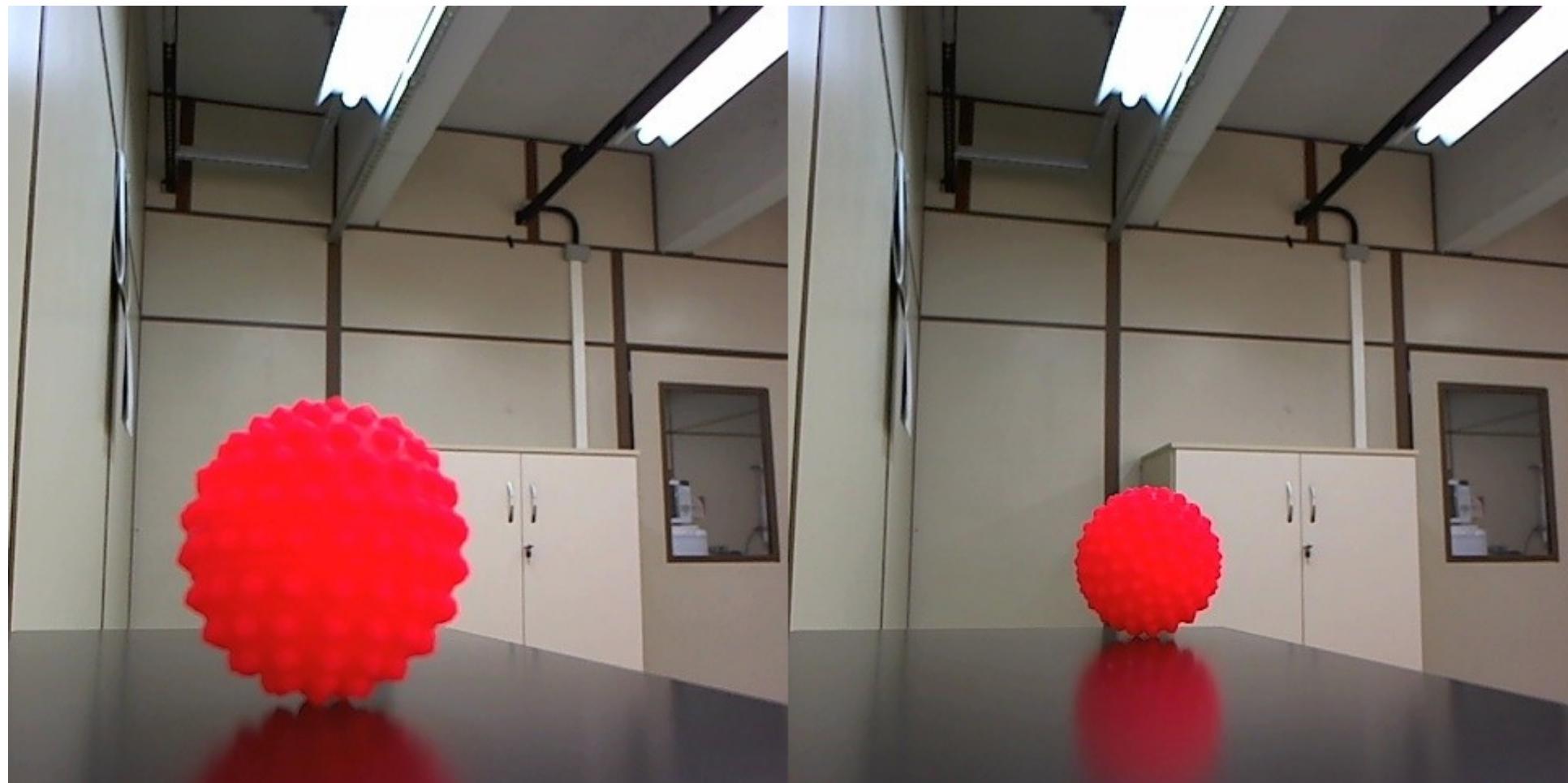
A imagem é influenciada...

- ... pelo tempo de exposição.



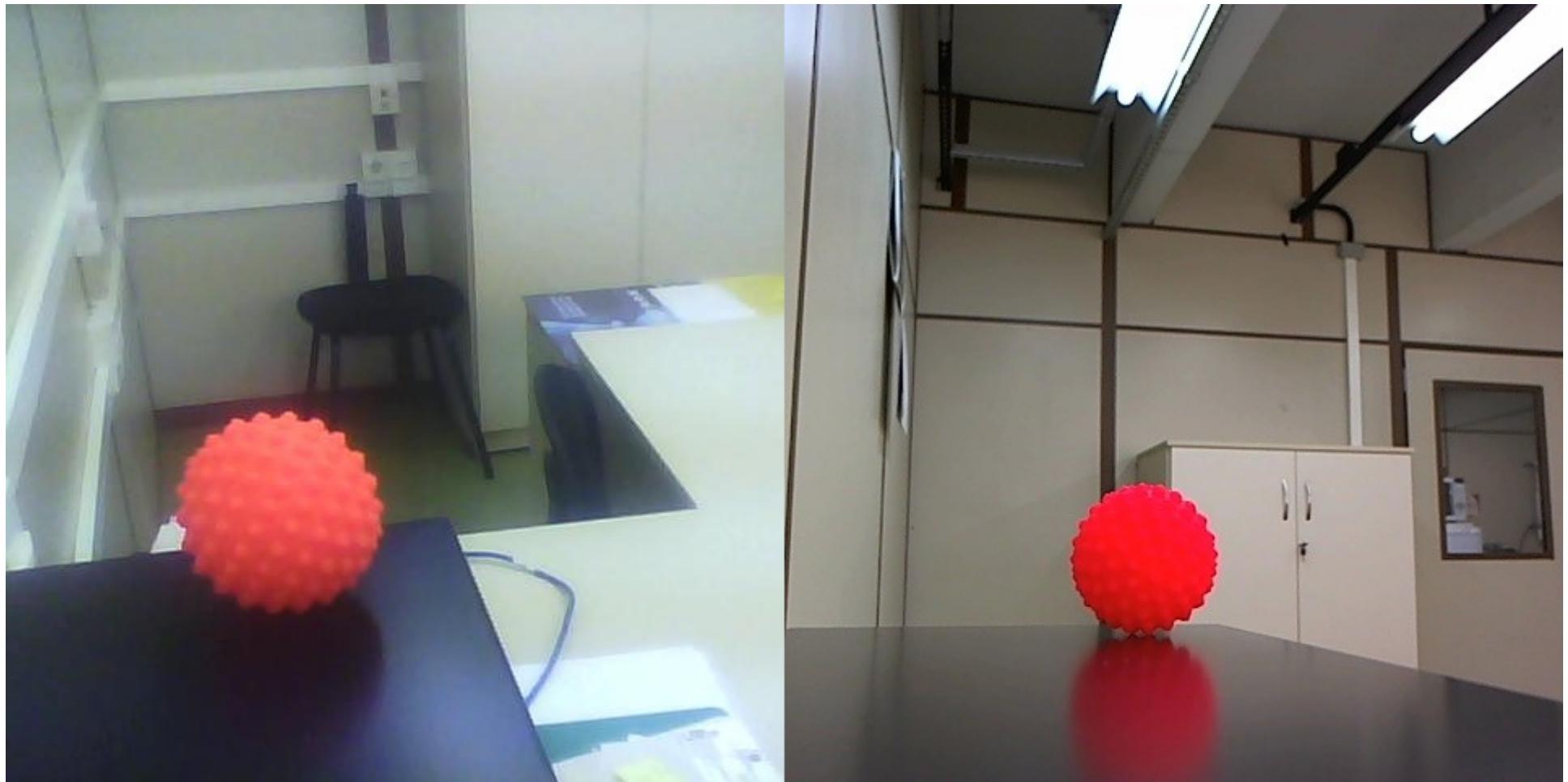
A imagem é influenciada...

- ... por variações de tamanho e distorções de perspectiva.



A imagem é influenciada...

- ... pelas características e pela qualidade da câmera.



Outros fatores

- Cor da luz.
 - A luz do Sol é *acromática*.
 - A luz de uma lâmpada incandescente é amarelada.
 - A luz do Sol filtrada por um vitral tem várias cores.
- Posição da luz.
 - Causa efeitos de sombra e reflexos.
- “Cor” do objeto.
 - = cor da luz refletida pelo objeto.
- Propriedades do material.
 - Afetam a forma como a luz é refletida (mesmo microscopicamente!).

Ferramentas

- Arquivos imagem.c e imagem.h.
 - Implementação em C de um subconjunto do formato bmp.
 - Prioriza clareza e facilidade de uso, não desempenho.

```
typedef struct
{
    int largura;
    int altura;
    int n_canais;
    float*** dados;
} Imagem;

Imagen* criaImagen (int largura, int altura, int n_canais);
void destroiImagen (Imagen* img);
Imagen* abreImagen (char* arquivo, int n_canais);
int salvaImagen (Imagen* img, char* arquivo);
Imagen* clonaImagen (Imagen* img);
void copiaConteudo (Imagen* in, Imagem* out);
```

Ferramentas

- Achar bugs ou comentários incorretos nos códigos do professor = bônus nas perguntas feitas em sala.
- Obs: as funções dadas **NÃO** testam exaustivamente os parâmetros recebidos.
 - Erros causados por parâmetros absurdos não serão considerados.

Vejamos alguns exemplos...

- Abrindo e salvando.
- Negativo de uma imagem em escala de cinza.
- Negativo de uma imagem colorida.
- Separando os canais de uma imagem colorida.
- Manipulando uma imagem em escala de cinza.
- Manipulando uma imagem colorida.