

Tópicos:

- ▶ Geometria da câmera e Projeção
- ▶ Calibração → conhecer os parâmetros geométricos de uma câmera
- ▶ Geometria epipolar → relação geométrica entre duas ou mais câmeras
- ▶ Estéreo binocular → obter profundidade a partir de duas imagens
- ▶ *Multiview* estéreo → reconstruir a partir de duas ou mais imagens
- ▶ *Structure From Motion (SFM)* → reconstruir e obter parâmetros de câmera a partir de duas ou mais imagens

Percepção do mundo: tridimensional

Superfícies planas: bidimensional

Projeção

- ▶ mundo → superfície plana



Noção de profundidade

- O chamado dos apóstolos Pedro e André (1308 – 1311, Duccio di Buoninsegna)



- Song Dynasty (século XII)



Noção de profundidade

- ▶ A confirmação das ordens dos Frades (1288-1292, Giotto)
 - ▶ Linhas paralelas convergem, mas não para um ponto em comum



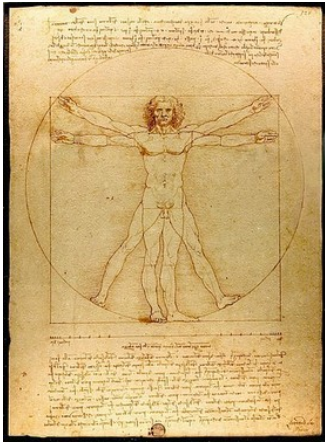
Uso de perspectiva linear

- O pagamento do tributo (1425, Masaccio)



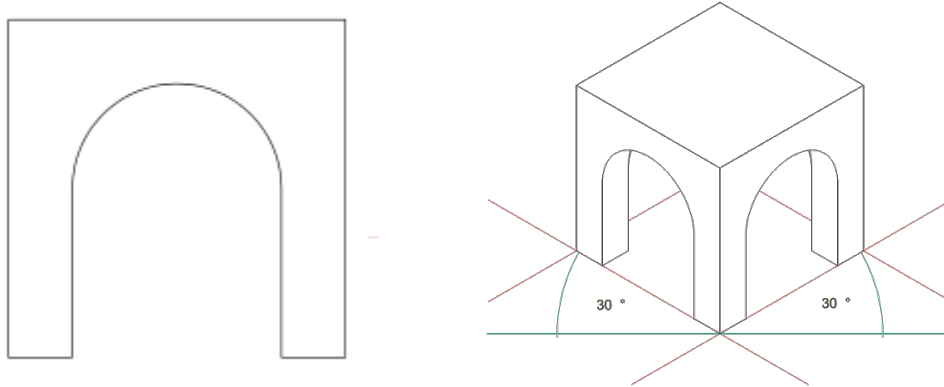
Renascença

- ▶ Estudo mais preciso nas artes, principalmente figura humana
- ▶ Filippo Brunelleschi
 - ▶ Métodos para desenho em perspectiva
 - ▶ <https://www.youtube.com/watch?v=u-OYhrilFo>



Tipos de Projeção

- ▶ O tipo de projeção influencia na interpretação da cena
- ▶ Cada projeção destaca certas características do objeto



Tipos de Projeção

- ▶ Paralela

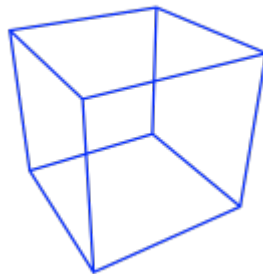
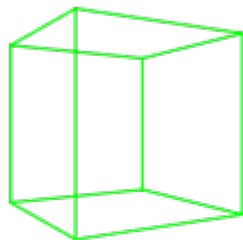
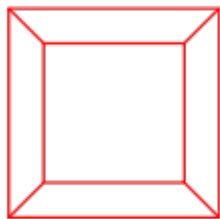
- ▶ Ortogonal (multiview)
- ▶ Axonométrica
- ▶ Oblíqua

- ▶ Perspectiva

- ▶ https://upload.wikimedia.org/wikipedia/commons/4/41/Graphical_projection_comparison.png

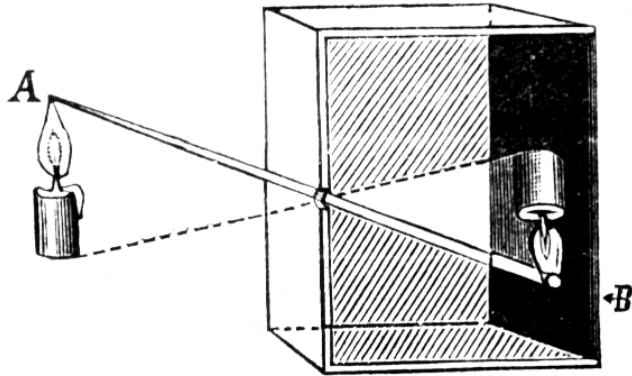
Projeção Perspectiva: ponto de fuga

- ▶ Retas paralelas, interceptam o plano de projeção?
 - ▶ Sim: convergem para um ponto de fuga
 - ▶ Não: não convergem
- ▶ Perspectiva com $\{1, 2, 3, \dots\}$ pontos de fuga



Projeção Perspectiva: origem

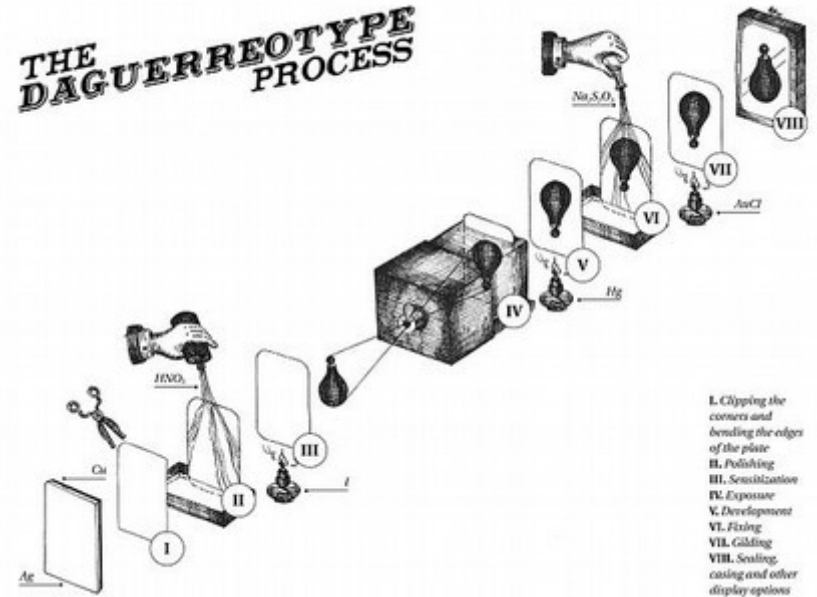
- ▶ Visão humana e câmeras possuem projeção perspectiva
- ▶ Câmera obscura
 - ▶ Centro de projeção (pinhole)
 - ▶ Plano Imagem (onde a imagem é projetada)



"Camera obscura 1" por Original: Fizyka z 1910Secondary: From Ru-Wiki. Initial description is/was here.. Licenciado sob Domínio público, via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Camera_obscura_1.jpg#/media/File:Camera_obscura_1.jpg

Projeção Perspectiva: origem

- ▶ **Daguerreótipo (1839)**
 - ▶ Primeiro processo fotográfico publicado



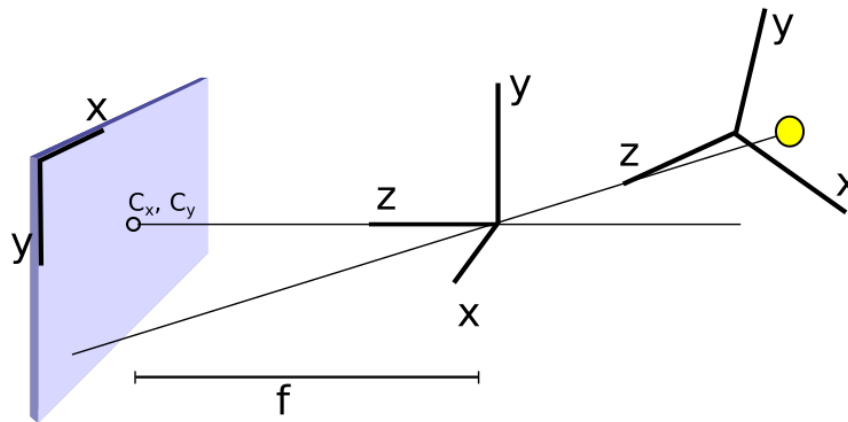
"Daguerreotype process" by This image has been created during "DensityDesign Integrated Course Final Synthesis Studio" at Polytechnic University of Milan, organized by DensityDesign Research Lab. Image is released under CC-BY-SA licence. Attribution goes to "Susanna Celeste Castelli, DensityDesign Research Lab". - Own work. Licensed under CC BY-SA 4.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Daguerreotype_process.jpg#/media/File:Daguerreotype_process.jpg

Formalização da câmera

Projeção Perspectiva: pin-hole

► Formalização da câmera pin-hole

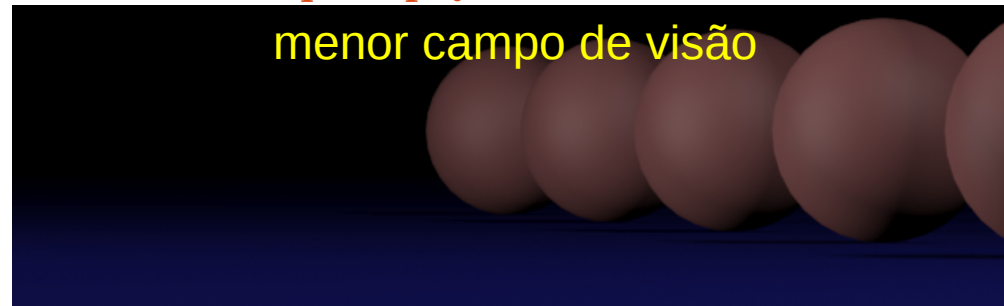
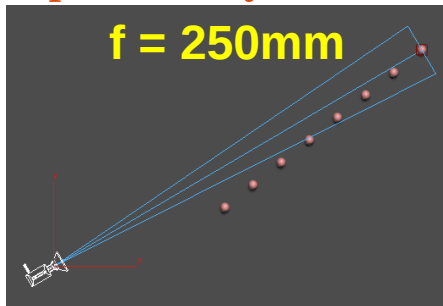
- Três sistemas de coordenadas:
 - Coordenadas de mundo
 - Coordenadas de câmera
 - Coordenadas de imagem (em pixels)
- O plano imagem pode ser colocado à frente do centro de projeção (pinhole)



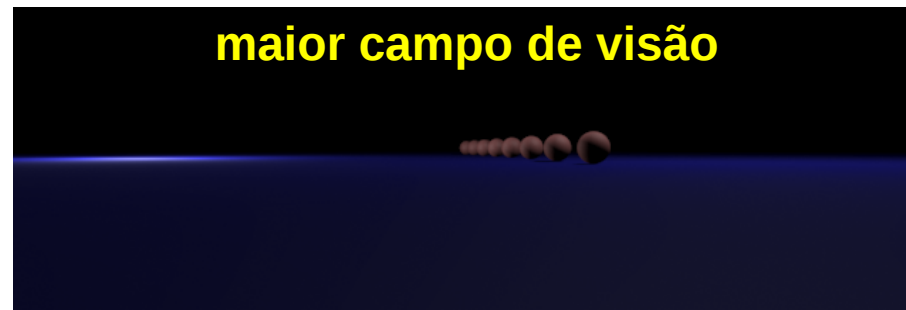
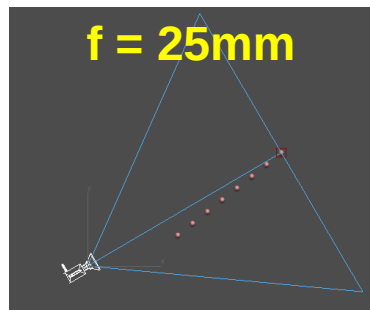
Projeção Perspectiva: campo de visão (FOV Field of View)

- ▶ Maior distância focal → menor campo de visão

<https://www.youtube.com/watch?v=IRq18WpQZC0>

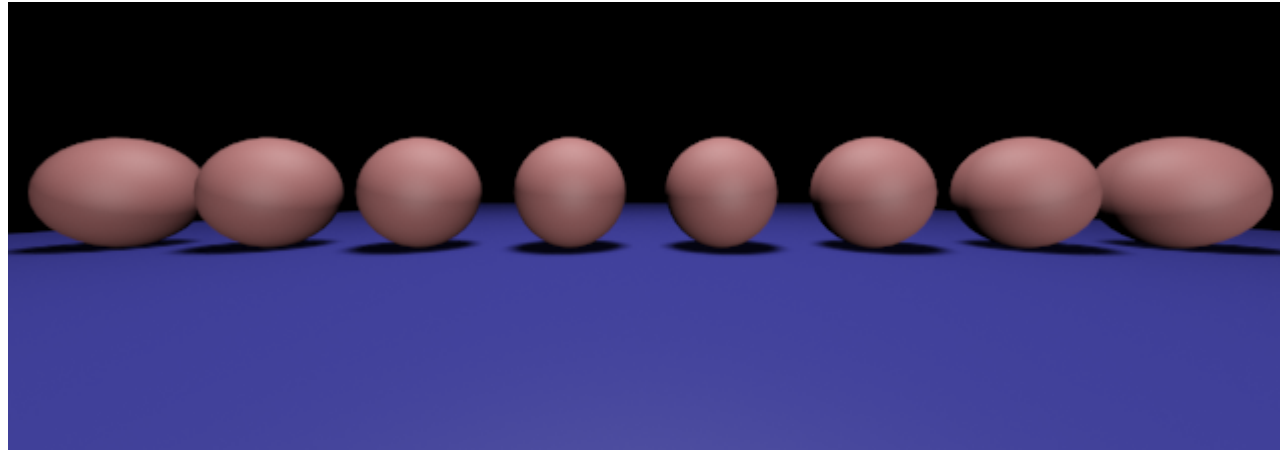
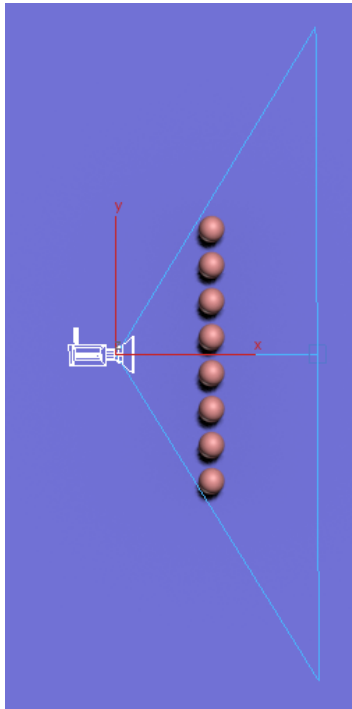


- ▶ Menor distância focal → maior campo de visão



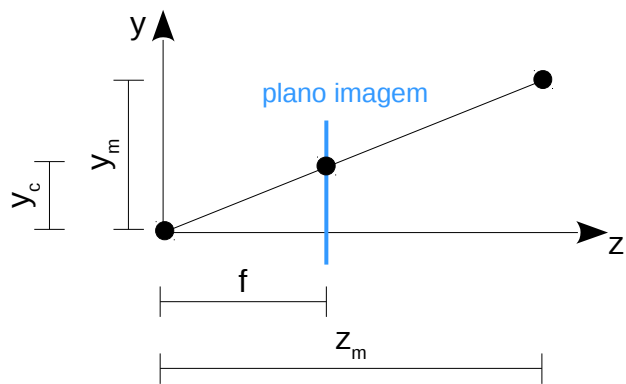
Projeção Perspectiva: distorção

- Efeito inerente à projeção perspectiva



Projeção Perspectiva

- Cálculo da projeção perspectiva



$$\begin{pmatrix} f x_c \\ f y_c \\ z_c \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix}$$

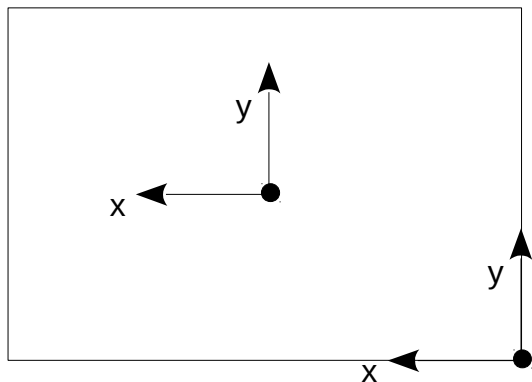
$$x_p = \frac{f x_c}{z_c} \quad y_p = \frac{f y_c}{z_c}$$

- Exercício

Calcule a projeção dos pontos $(3, 5, 2)$ e $(6, 10, 4)$ no plano imagem considerando $f = 1$

Projeção Perspectiva

- Cálculo da projeção perspectiva



$$\begin{pmatrix} fx_c + z_cc_x \\ fy_c + z_cc_y \\ z_c \end{pmatrix} = \begin{pmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix}$$

$$x_p = \frac{fx_c}{z_c} + c_x \quad y_p = \frac{fy_c}{z_c} + c_y$$

- Exercício

Calcule a projeção dos pontos (3, 5, 2) no plano imagem considerando $f = 1$ e $(c_x = 6, c_y = 4)$

Matriz Intrínsecos e Extrínsecos

- ▶ Matriz de intrínsecos

$$\begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

- ▶ Matriz de extrínsecos $[R \mid t]$

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix}$$

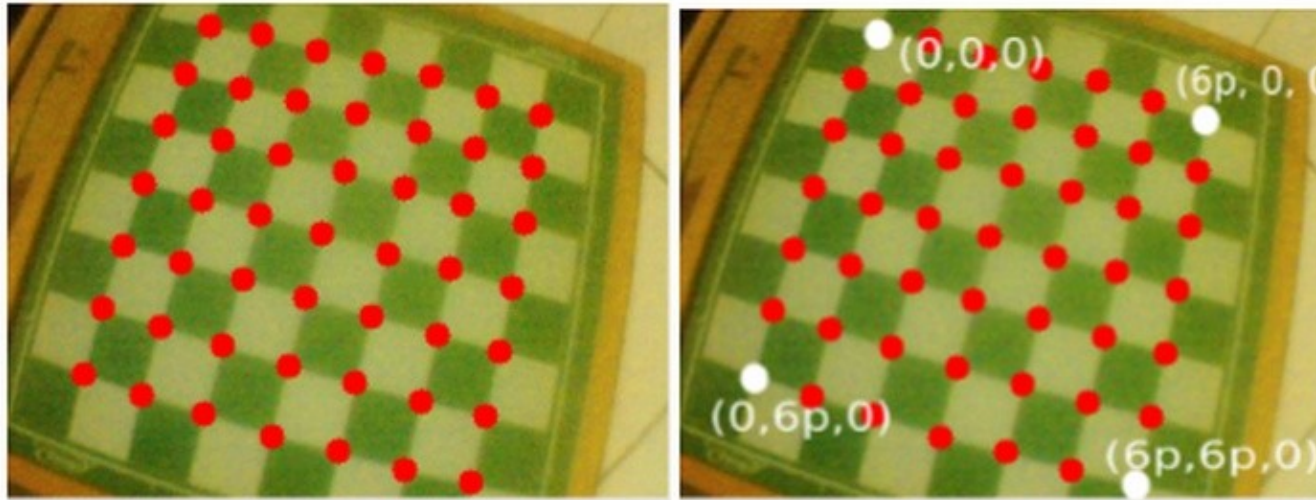
Matriz de Câmera

- ▶ Produto $M_{\text{int}} M_{\text{ext}}$
- ▶ Obs.: há variações quanto à forma dessas matrizes em decorrência das convenções adotadas

Calibração

Metodologia para Calibração Manual

- ▶ Pontos conhecidos em coordenadas de mundo são identificados na imagem
 - ▶ Comum utilizar um tabuleiro de xadrez
- ▶ Associar coordenada de mundo \leftrightarrow coordenada de imagem
 - ▶ Exemplo: $(0, 0, 0) \leftrightarrow (208, 21)$



Metodologia para Calibração Manual

- ▶ 1º passo: achar os cantos do tabuleiro
 - ▶ `findChessBoardCorners`
- ▶ 2º passo: aplicar um método de calibração, exemplo:
 - ▶ DLT: direct linear transformation
 - ▶ Método de Zhang (disponível no OpenCV)

Método DLT (*Direct Linear Transformation*)

- ▶ Considere os pontos de mundo P_i e suas projeções $p_i = (x_i, y_i, 1)$:

$$\lambda_i p_i = M P_i$$

- ▶ Seja M expresso como:

$$M = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} \quad m_1 \rightarrow 1^{\text{a}} \text{ linha de } M \text{ (1x4)}$$

- ▶ Então:

$$\begin{aligned} P_i^T m_1^T - \lambda_i x_i &= 0 \\ P_i^T m_2^T - \lambda_i y_i &= 0 \\ P_i^T m_3^T - \lambda_i &= 0 \end{aligned} \quad \begin{bmatrix} P_i^T & 0 & 0 & -x_i \\ 0 & P_i^T & 0 & -y_i \\ 0 & 0 & P_i^T & -1 \end{bmatrix} \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \\ \lambda_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Método DLT (*Direct Linear Transformation*)

- ▶ Colocando em uma única matriz todos os pontos, obtemos:

$$\begin{bmatrix} P_1^T & 0 & 0 & -x_1 & 0 & 0 & \dots \\ 0 & P_1^T & 0 & -y_1 & 0 & 0 & \dots \\ 0 & 0 & P_1^T & -1 & 0 & 0 & \dots \\ P_2^T & 0 & 0 & 0 & -x_2 & 0 & \dots \\ 0 & P_2^T & 0 & 0 & -y_2 & 0 & \dots \\ 0 & 0 & P_2^T & 0 & -1 & 0 & \dots \\ P_3^T & 0 & 0 & 0 & 0 & -x_3 & \dots \\ 0 & P_3^T & 0 & 0 & 0 & -y_3 & \dots \\ 0 & 0 & P_3^T & 0 & 0 & -1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- ▶ São necessários ao menos 6 pontos para ter mais equações que incógnitas
- ▶ Resolve-se por mínimos quadrados

Visão Estéreo

O que é profundidade?

Como percebemos a profundidade?



Como percebemos a profundidade?

- ▶ Perspectiva
- ▶ Oclusão
- ▶ Conhecimento prévio dos objetos
- ▶ Posição relativa ao horizonte
- ▶ Iluminação
- ▶ Paralaxe
- ▶ **Visão binocular**



By Andrzej Barabasz (Chepyr) - Own work, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=143793>

Visão estéreo

- ▶ *stereós* (sólido, firme, tridimensional)
- ▶ Estereopsia (-opsis: aparência): percepção de tridimensionalidade de um sólido
 - ▶ Nos seres humanos em decorrência principalmente da visão binocular

Estereoscopia

- ▶ *skopeco*: observar
- ▶ Formas artificiais para criar ilusão de profundidade:
 - ▶ Estereoscópio
 - ▶ Anáglifo
 - ▶ HMD (*head-mounted display*)
 - ▶ *Shutter Glasses*
 - ▶ Óculos com lentes polarizadas
 - ▶ Autoestereograma

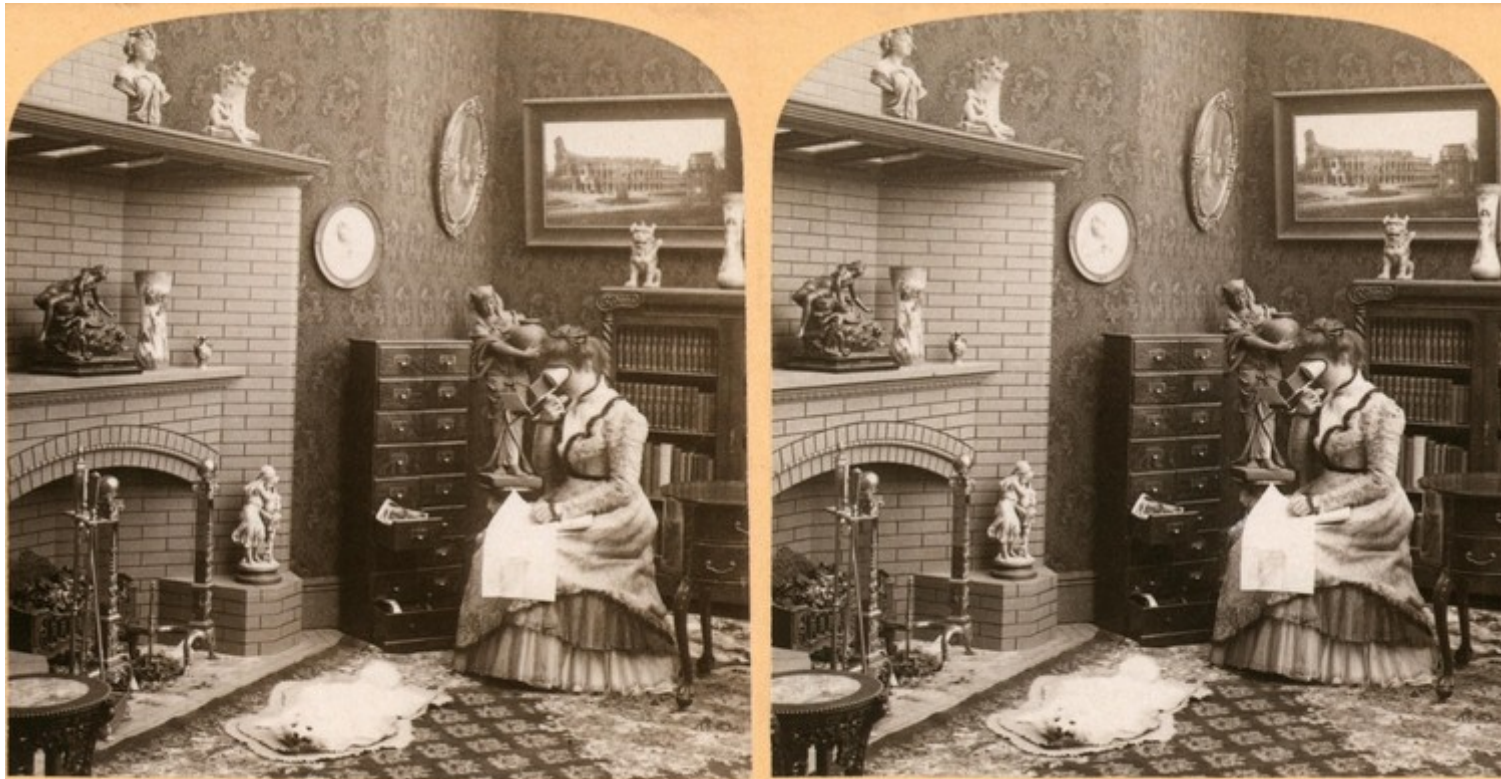
Estereograma

- ▶ Imagem utilizada para propósitos de estereoscopia

Estereoscópio



Estereoscópio



Invertido em relação ao original para visualizar com os olhos cruzados

Anáglifo



HMD (Head Mounted Display)



"Oculus Rift - Developer Version - Back and Control Box" by Sebastian Stabinger - Own work. Licensed under CC BY 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Oculus_Rift_-_Developer_Version_-_Back_and_Control_Box.jpg#/media/File:Oculus_Rift_-_Developer_Version_-_Back_and_Control_Box.jpg

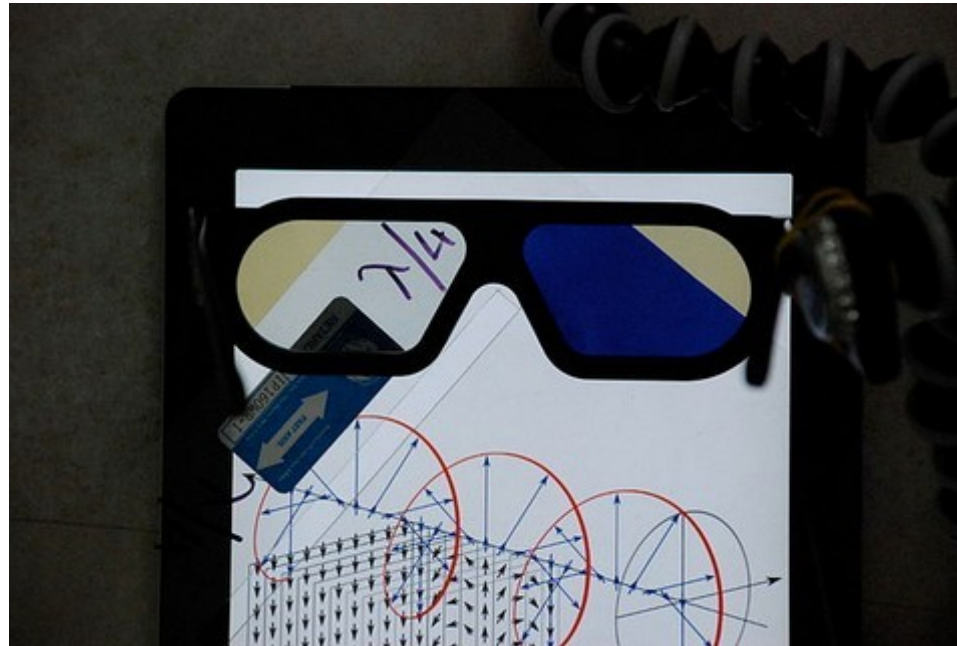
Shutter Glasses

- Bloqueia a visão em uma das lentes de forma alternada e em sincronia com o que é exibido em um display externo

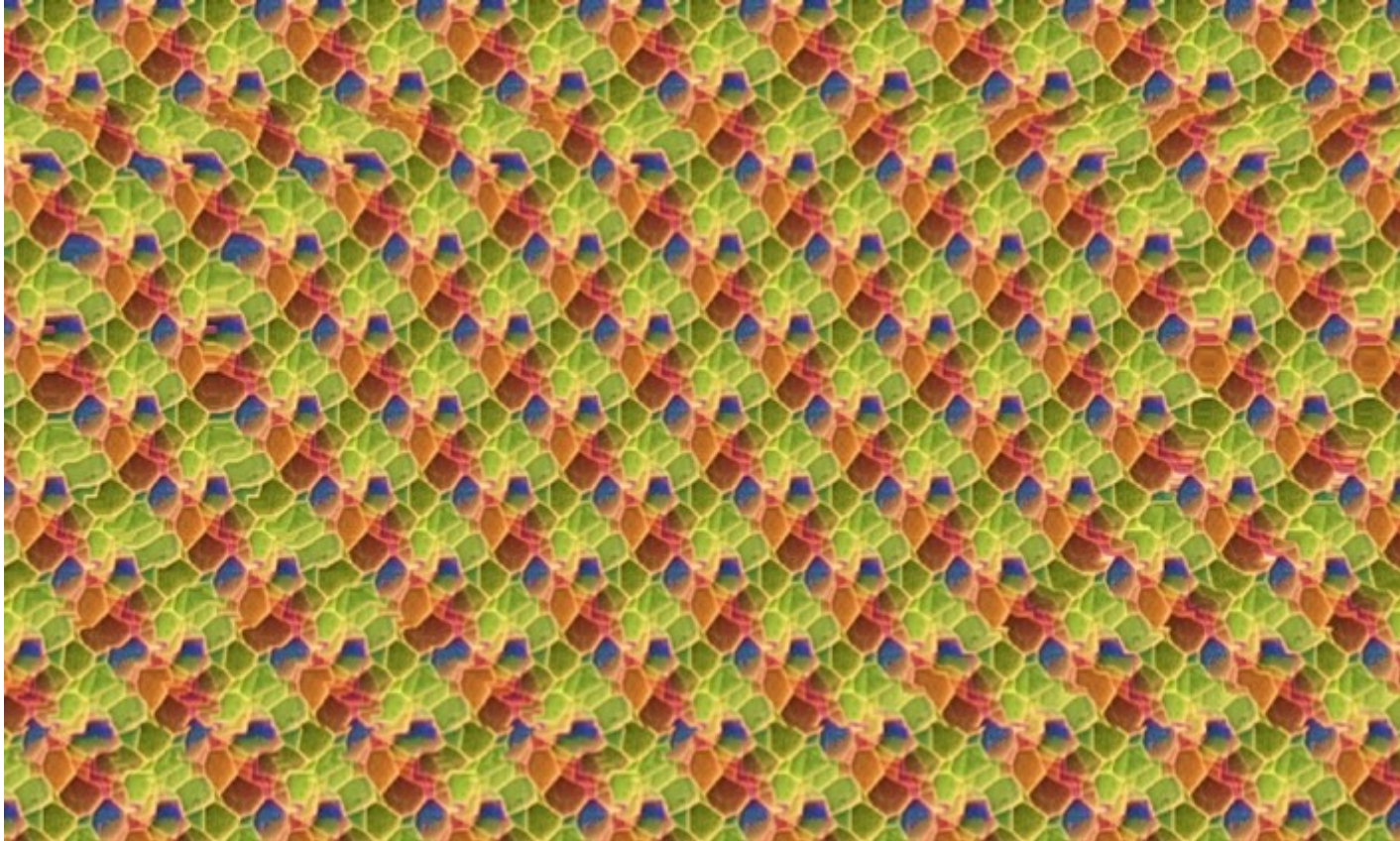


Lentes poralizadas

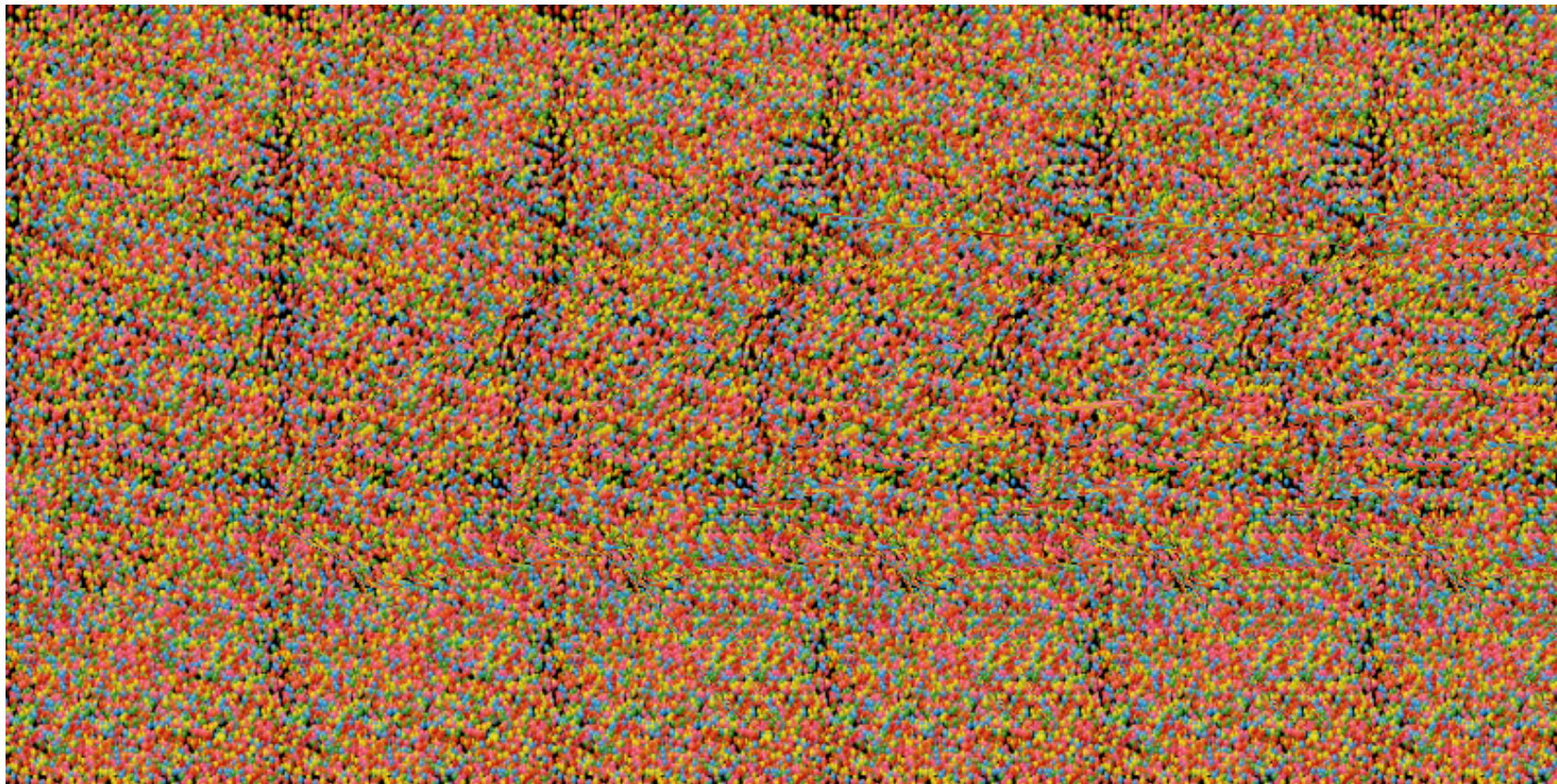
- ▶ Cada lente bloqueia as ondas EM com determinada polarização



Autoestereograma



Autoestereograma

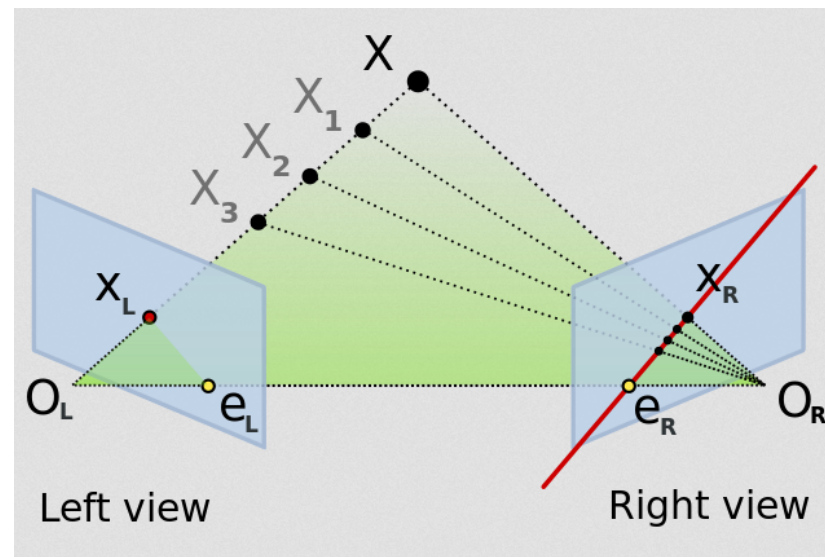


Visão Estéreo Computacional

- ▶ Geometria Epipolar: baseada no modelo de câmera pin-hole

- ▶ Variáveis:

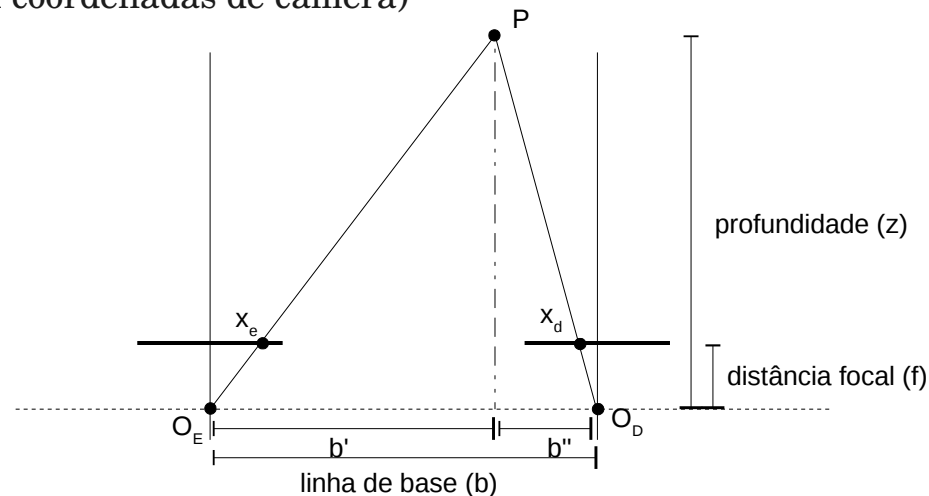
- ▶ Pontos epipolares (e_L, e_R)
- ▶ Plano epipolar (X, O_L, O_R)
- ▶ Linha epipolar ($X_L \rightarrow e_L, X_R \rightarrow e_R$)
- ▶ Se X_L é a projeção tanto de X como de X_1, X_2, \dots
- ▶ Como encontrar a projeção de X à direita?
 - ▶ Usa-se a restrição epipolar: X_R está na linha epipolar



"Epipolar geometry" by Arne Nordmann (norro) - Own work (Own drawing). Licensed under CC BY-SA 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Epipolar_geometry.svg#/media/File:Epipolar_geometry.svg

Visão Estéreo Computacional

- ▶ Restrições quem facilitam a resolução do problema
 - ▶ Os dois eixos ópticos estão paralelos e as câmeras no mesmo nível $\rightarrow y_e = y_d$
- ▶ A profundidade pode ser obtida por triangulação
 - ▶ Observe que $x_e > x_d$ (ambas em coordenadas de câmera)
 - ▶ Disparidade: $x_e - x_d$



$$z = f \frac{b}{x_e - x_d}$$

Visão Estéreo Computacional

$$z = f \frac{b}{x_e - x_d}$$

- ▶ O que acontece com a disparidade quando o objeto está mais próximo ou mais distante?
- ▶ Profundidade é inversamente proporcional à disparidade
- ▶ Simplificação prática:
 - ▶ armazena-se apenas a disparidade em coordenadas de imagem ao invés de z
 - ▶ valor de disparidade entre $[0, 255]$ para propósitos de visualização



Visão Estéreo Computacional

► Problema da disparidade estéreo:

- Dado uma projeção x_e , como encontrar a projeção x_d ?
- Geralmente resolve-se com um algoritmo de *matching*

► Algoritmos de Matching:

- Baseados em área (denso):
 - **Mais comum**
 - Disparidade para **todos** os pixels da imagem (geralmente da esquerda)
 - Para cada pixel da imagem esquerda, encontrar o correspondente na imagem direita
- Baseados em elementos (esparso):
 - Disparidade calculada somente para alguns elementos da imagem

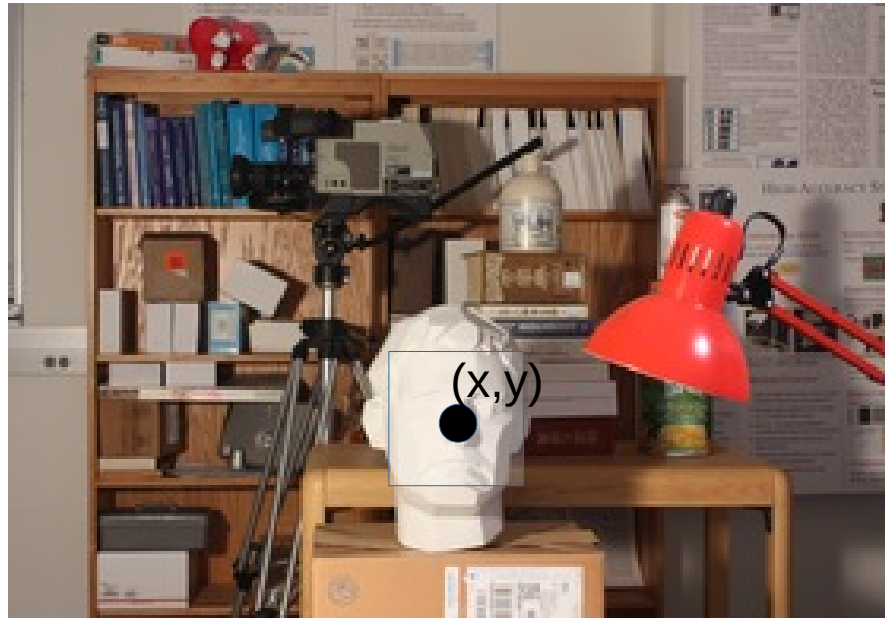
Visão Estéreo Computacional: matching baseado em área



Visão Estéreo Computacional: matching baseado em área

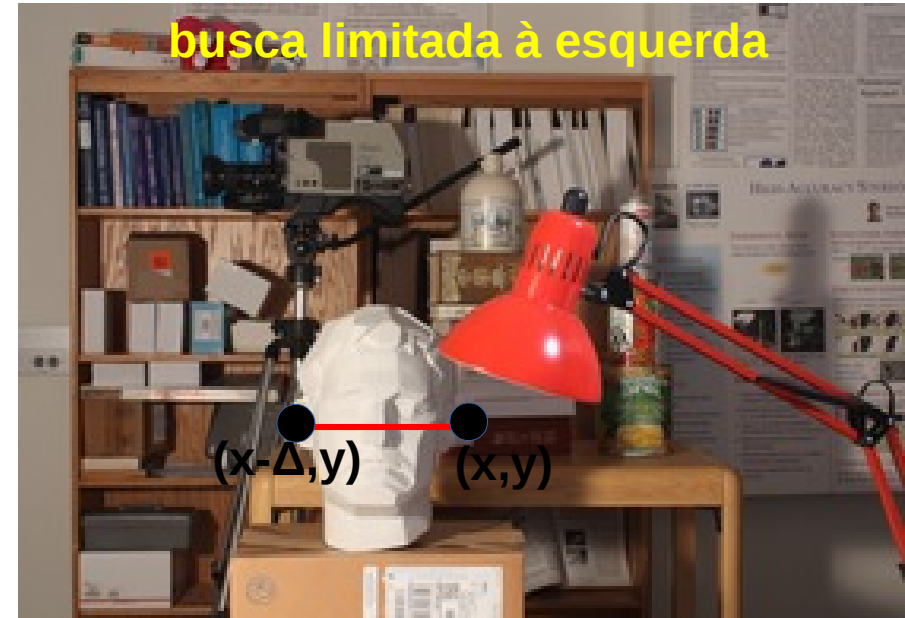


Visão Estéreo Computacional: matching baseado em área



mínimo (ex: SAD, SSD)

Visão Estéreo Computacional: matching baseado em área

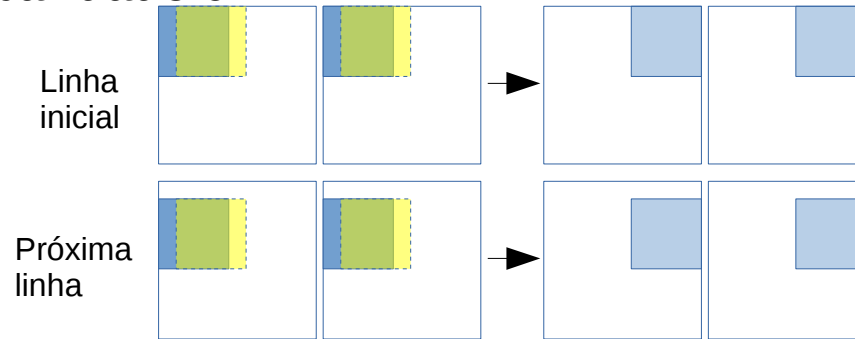


Visão Estéreo Computacional: matching baseado em área

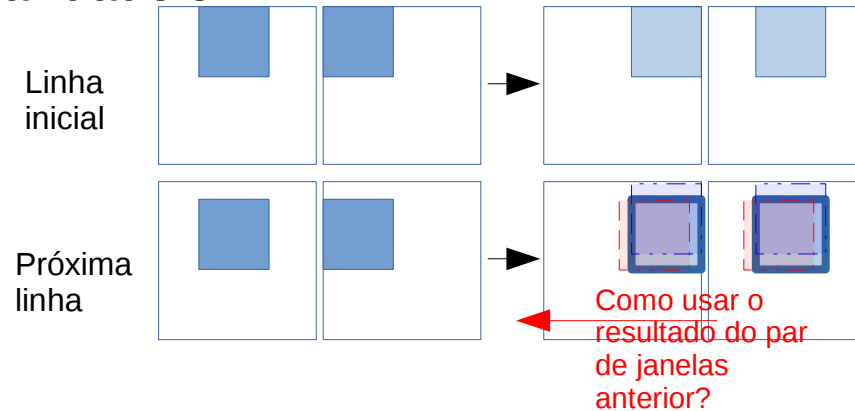
- ▶ Otimizações se a métrica for SSD/SAD/similares
- ▶ Para cada disparidade $d = \{0, 1, 2, 3, \dots, d_{\max}\}$
 - ▶ Atribua y um valor inicial (respeitando os limites da janela)
 - Posicionar uma janela em $I_e(x_0+d, y)$ e outra em $I_d(x_0, y)$, calcular a função e armazenar d em $D(x_0+d, y)$ como melhor solução caso assim tenha sido
 - Deslocar ambas as janelas para a direita e fazer uma nova verificação enquanto for possível deslocar (como usar o resultado anterior para otimizar esse passo?)
 - ▶ Incremente y e repita o procedimento anterior (como usar o resultado da linha anterior para otimizar o resultado da linha atual?)

Visão Estéreo Computacional: matching baseado em área

Disparidade 0



Disparidade 5



Como usar o resultado do par de janelas da linha anterior?

Visão Estéreo Computacional: matching baseado em PD

- ▶ Técnica na elaboração de um algoritmo: resolve um problema em função da solução de subproblemas
- ▶ Explorada na década de 50 por Richard Bellman (origem do nome da técnica)
- ▶ Resolvem de forma eficiente problemas de otimização combinatorial (geralmente $n!$ ou m^n)

Visão Estéreo Computacional: matching baseado em PD

- ▶ Exemplo: sequência de fibonacci

1, 1, 2, 3, 5, 8, 13, 21, ...

- ▶ os dois primeiros termos são iguais a 1
- ▶ cada termo seguinte é igual à soma dos dois anteriores

- ▶ O n-ésimo termos da sequência pode ser expressa como uma função recursiva:

$\text{fib}(1) = 1$ (caso base)

$\text{fib}(2) = 1$ (caso base)

$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$ para $n > 2$

- ▶ Observe que expressamos um problema em função de subproblemas:

$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$ para $n > 2$



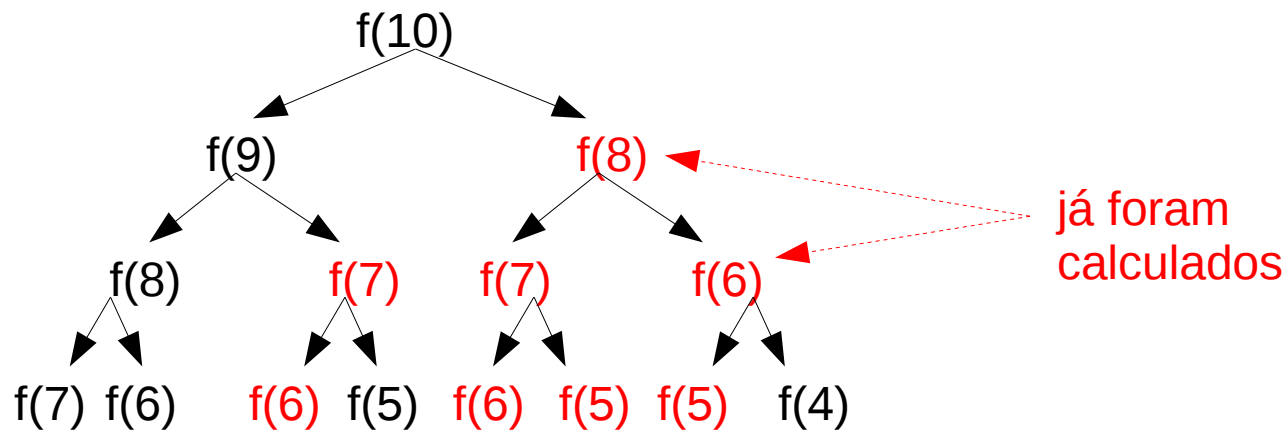
problema subproblemas

Visão Estéreo Computacional: matching baseado em PD

- ▶ Solução para evitar resolver o mesmo subproblema mais de uma vez: *memoization*
 - ▶ Ao encontrar a solução para um subproblema com n parâmetros $(p_1, p_2, p_3, \dots, p_n)$, armazena-se em uma matriz n -dimensional $M[p_1][p_2]..[p_n]$ seu resultado
 - ▶ Inicialmente essa matriz M possui algum valor que indica que aquele subproblema com aqueles parâmetros ainda não foi calculado (-1, por exemplo)
 - ▶ Ao precisar da solução de um subproblema, verifica-se se já foi calculado e em caso afirmativo, a solução do subproblema é recuperado na matriz
- ```
int fib(int n) {
 if(n <= 2) return 1;
 else if(M[n] != -1) return M[n];
 else return fib(n-1) + fib(n-2)
}
```

## Visão Estéreo Computacional: matching baseado em PD

- ▶ Cada subproblema envolve a resolução de outros subproblema (a não ser o caso base):  
 $\text{fib}(10) = \text{fib}(9) + \text{fib}(8)$   
 $\text{fib}(9) = \text{fib}(8) + \text{fib}(7)$   
 $\text{fib}(8) = \text{fib}(7) + \text{fib}(6)$
- ▶ Esta solução recursiva é custosa

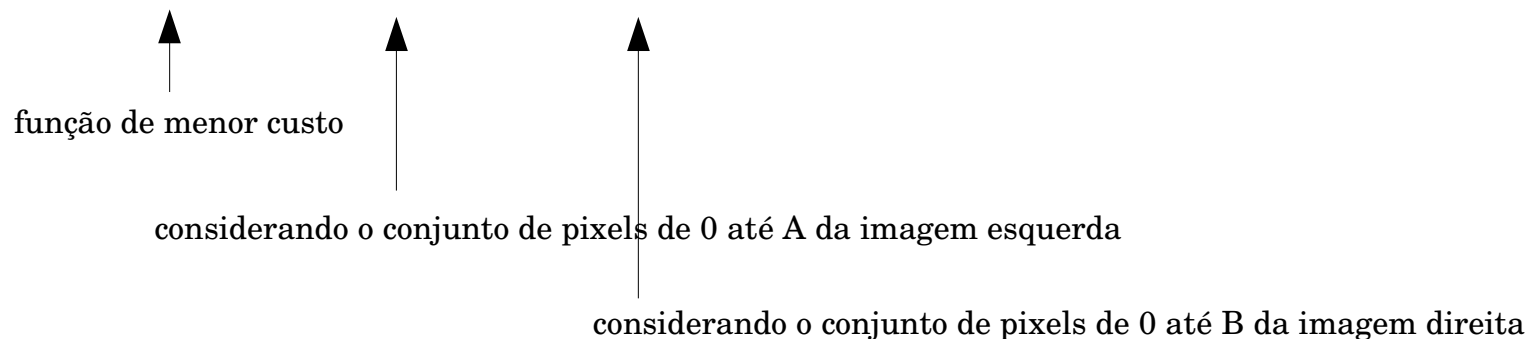


## Visão Estéreo Computacional: matching baseado em PD

- ▶ Solução para evitar resolver o mesmo subproblema mais de uma vez: *memoization*
  - ▶ Ao encontrar a solução para um subproblema com  $n$  parâmetros  $(p_1, p_2, p_3, \dots, p_n)$ , armazena-se em uma matriz  $n$ -dimensional  $M[p_1][p_2]..[p_n]$  seu resultado
  - ▶ Inicialmente essa matriz  $M$  possui algum valor que indica que aquele subproblema com aqueles parâmetros ainda não foi calculado (-1, por exemplo)
  - ▶ Ao precisar da solução de um subproblema, verifica-se se já foi calculado e em caso afirmativo, a solução do subproblema é recuperado na matriz
- ```
int fib(int n) {  
    if(n <= 2)      return 1;  
    else if(M[n] != -1) return M[n];  
    else    return fib(n-1) + fib(n-2)  
}
```

Visão Estéreo Computacional: matching baseado em PD

- ▶ Cada linha (relativa a linhas epipolares com n pixels) é resolvida independentemente:
 $E(i)$: i -ésimo pixel da imagem esquerda
 $D(j)$: j -ésimo pixel da imagem direita
- ▶ Deseja-se casar os pixels da imagem esquerda com os da imagem direita com o menor custo total possível:
 $f(\{0, 1, 2, \dots, A\}, \{0, 1, 2, \dots, B\}) = ?$



- ▶ O objetivo é encontrar o menor custo considerando todos os pixels:
 $f(\{0, \dots, n-1\}, \{0, \dots, n-1\})$

Visão Estéreo Computacional: matching baseado em PD

- ▶ Resolvendo em função de subproblemas:

$f(\{0, \dots, i\}, \{0, \dots, j\}) = \text{custo casar } E(i) \text{ e } D(j) +$

$\text{MIN}(\quad f(\{0, \dots, i-1\} \quad , \{0, \dots, j-1\}),$

$f(\{0, \dots, i\} , \{0, \dots, j-1\}),$

$f(\{0, \dots, i-1\} \quad , \{0, \dots, j\})$

3 subproblemas, onde:

nem i, nem j podem ser recasados

i pode ser recasado

j pode ser recasado

- ▶ Como o problema pode ser caracterizado em função de duas variáveis, o memoization é aplicado utilizando duas matrizes $n \times n$, onde:

- ▶ $C[i][j]$ representa o custo $f(\{0, \dots, i\}, \{0, \dots, j\})$

- ▶ $P[i][j]$ representa a partir de quais subconjuntos foi obtido o menor custo (recuperar a disparidade)

- ▶ Subproblemas estão armazenados em células vizinhas da matriz C , $P[i][j]$ pode ser:

- UP: $f(\{0, \dots, i-1\} \quad , \{0, \dots, j\})$

- LEFT: $f(\{0, \dots, i\} , \{0, \dots, j-1\})$

- UPLEFT: $f(\{0, \dots, i-1\} \quad , \{0, \dots, j-1\})$

Visão Estéreo Computacional: matching baseado em PD

- ▶ Os casos bases são:

$f(\{0\}, \{0\}) = \text{custo casar } E(0) \text{ e } D(0)$

$f(\{0, 1, 2, \dots, i\}, \{0\}) = \text{custo casar } E(i) \text{ e } D(0) + f(\{0, 1, 2, \dots, i-1\}, \{0\})$

- ▶ Observe que como um pixel da imagem esquerda casa necessariamente com um pixel da imagem direita que está à esquerda, considere apenas $f(\{0, \dots, i\}, \{0, \dots, j\})$ onde $i \geq j$
- ▶ Percorrer a matriz triangular inferior de C (pois $i \geq j$) indo de $[0][0]$ até $[n][n]$ evita que você tenha que conferir se a função f já foi calculada, pois é garantido, nesse caso, que já tenha sido
- ▶ A disparidade é recuperada percorrendo-se a matriz C de $C[n][n]$ até $C[0][0]$, seguindo a direção dada pela matriz P

Visão Estéreo Computacional: matching baseado em Otimização

- ▶ Modelo de Ising: explica o comportamento de partículas
- ▶ Uma malha $n \times n$ partículas, cada qual com carga -1 ou 1
- ▶ Cada partícula contribui na energia total do sistema em função da diferença de carga com seus vizinhos

$$E = -J \sum_{x_i, x_j} x_i x_j$$

- ▶ $E(c)$ representa a energia total do sistema na configuração c

Visão Estéreo Computacional: matching baseado em Otimização

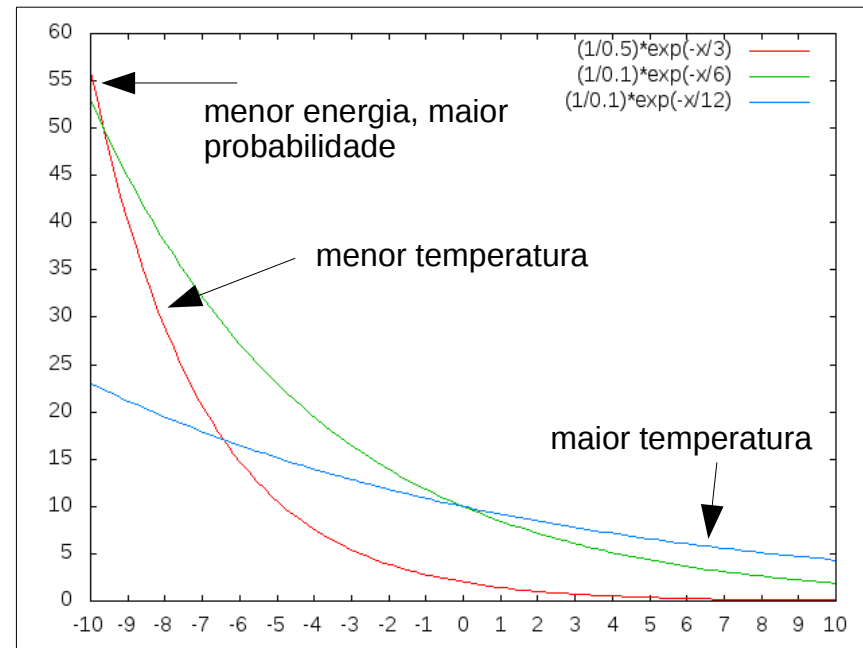
- ▶ A probabilidade de uma dada configuração é dada por:
 - ▶ No gráfico ao lado, y não está normalizado

$$P(c') = \frac{1}{Z} e^{\frac{-E(c')}{kT}}$$

↑ função de normalização

↑ constante


↑ temperatura do sistema



Visão Estéreo Computacional: matching baseado em Otimização

► Algoritmo de metrópolis usando o modelo de Ising

- Estando em uma configuração do sistema x , propõe-se uma nova configuração x' (incremento/decremento de disparidade de um pixel)
- Se a mudança acarreta em uma menor energia do sistema, aceita-se o novo estado
- Caso contrário, a probabilidade de aceitar a nova configuração é:

$$\alpha(x'|x) = e^{-2J(\frac{\sum_{p' \in V(p)} |p-p'|}{8})}$$


probabilidade (alpha) é maior para pixels similares aos vizinhos

- Resultado: <https://www.youtube.com/watch?v=YgNE9M1pJCM>

Visão Estéreo Computacional: matching baseado em Otimização

- ▶ Outra aplicação de Metrópolis + Ising: remoção de ruídos

