Classroom Examples of Robustness Problems in Geometric Computations

L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, C. Yap

Rafael C. S. Schouery
schouery@gmail.com

Introdução

Introdução

- **Geometria Computacional**: Utiliza RAM.

Introdução

- **Geometria Computacional**: Utiliza RAM.
- **Computadores**: Utilizam ponto flutuante.

Introdução

- **Geometria Computacional**: Utiliza RAM.
- **Computadores**: Utilizam ponto flutuante.
- Essa diferença cria espaço para falhas.

Introdução

- **Geometria Computacional**: Utiliza RAM.
- **Computadores**: Utilizam ponto flutuante.
- Essa diferença cria espaço para falhas.
- Não há papers sobre o que pode dar errado.

Problemas

Problemas

- **Professores:** pouco material sobre problemas.

Problemas

- **Professores:** pouco material sobre problemas.
- **Alunos:** subestimam o que pode dar errado.

Problemas

- **Professores:** pouco material sobre problemas.
- **Alunos:** subestimam o que pode dar errado.
- **Pesquisadores:** acham que soluções simples resolvem o problema.

Objetivos

- Dar exemplos sobre o que pode acontecer
- Explicar por ocorrem
- Analisar o algoritmo incremental de fecho convexo

Regras para o Experimento

- Código em C++.
- Números são double.
- Exemplos em decimal, mas a representação é exata na máquina.

Ponto Flutuante

Ponto Flutuante

- Padrão IEEE784

Ponto Flutuante

- Padrão IEEE784
- Formato: $\pm m 2^e$

Ponto Flutuante

- Padrão IEEE784
- Formato: $\pm m2^e$
- $m = 1.m_1m_1...m_{52}$

Ponto Flutuante

- Padrão IEEE784
- Formato: $\pm m 2^e$
- $m = 1.m_1 m_1 ... m_{52}$
- $-1024 < e < 1024$

Ponto Flutuante

- Padrão IEEE784
- Formato: $\pm m2^e$
- $m = 1.m_1m_1...m_{52}$
- $-1024 < e < 1024$
- Arredondado sistematicamente

Orientação

Orientação

- orientation(p,q,r) =
  sign((q$_x$ - p$_x$)(r$_y$ - p$_y$) - (q$_y$ - p$_y$)(r$_x$ - p$_x$))

Orientação

- orientation(p,q,r) =
  sign($(q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)$)
- esquerda: 1
  direita: -1
  colinear: 0

Orientação

- orientation(p,q,r) =
  sign((q$_x$ - p$_x$)(r$_y$ - p$_y$) - (q$_y$ - p$_y$)(r$_x$ - p$_x$))
- esquerda: 1
  direita: -1
  colinear: 0
- float_orient(p,q,r): implementação de orientation(p,q,r) usando float

Possíveis Erros

Possíveis Erros

- **Arredondar para zero:** Encontramos 0 ao invés de 1 ou -1

Possíveis Erros

- **Arredondar para zero:** Encontramos 0 ao invés de 1 ou -1
- **Perturbação do zero:** Encontramos 1 ou -1 ao invés de 0

Possíveis Erros

- **Arredondar para zero:** Encontramos 0 ao invés de 1 ou -1
- **Perturbação do zero:** Encontramos 1 ou -1 ao invés de 0
- **Inversão de sinal:** Encontramos 1 ou invés de -1 e vice-versa.

Experimento: float_orient

Experimento: float_orient

- Escolhemos três pontos: p, q e r e valor u

Experimento: float_orient

- Escolhemos três pontos: p, q e r e valor u
- float_orient((p$_x$ + xu, p$_y$ + yu), q, r)
  0 ≤ x, y ≤ 255

Experimento: float_orient

- Escolhemos três pontos: p, q e r e valor u
- float_orient((p$_x$ + xu, p$_y$ + yu), q, r)
  0 ≤ x, y ≤ 255
- u é o incremento para o próximo float

Experimento: float_orient

- Escolhemos três pontos: p, q e r e valor u
- float_orient((p$_x$ + xu, p$_y$ + yu), q, r)
  0 ≤ x, y ≤ 255
- u é o incremento para o próximo float
- Exemplo: u = 2^{-53} se p$_x$ = 0.5

Resultado

- Matriz com células para cada valor de x e y
- Pontos Amarelos são colineares
- Pontos Vermelhos estão à direita
- Pontos Azuis estão à esquerda
- Linha preta: aproximação do segmento (q,r)

1° Experimento

- q = (12, 12)
- r = (24, 24)
- p = (0.5, 0.5)

1° Experimento

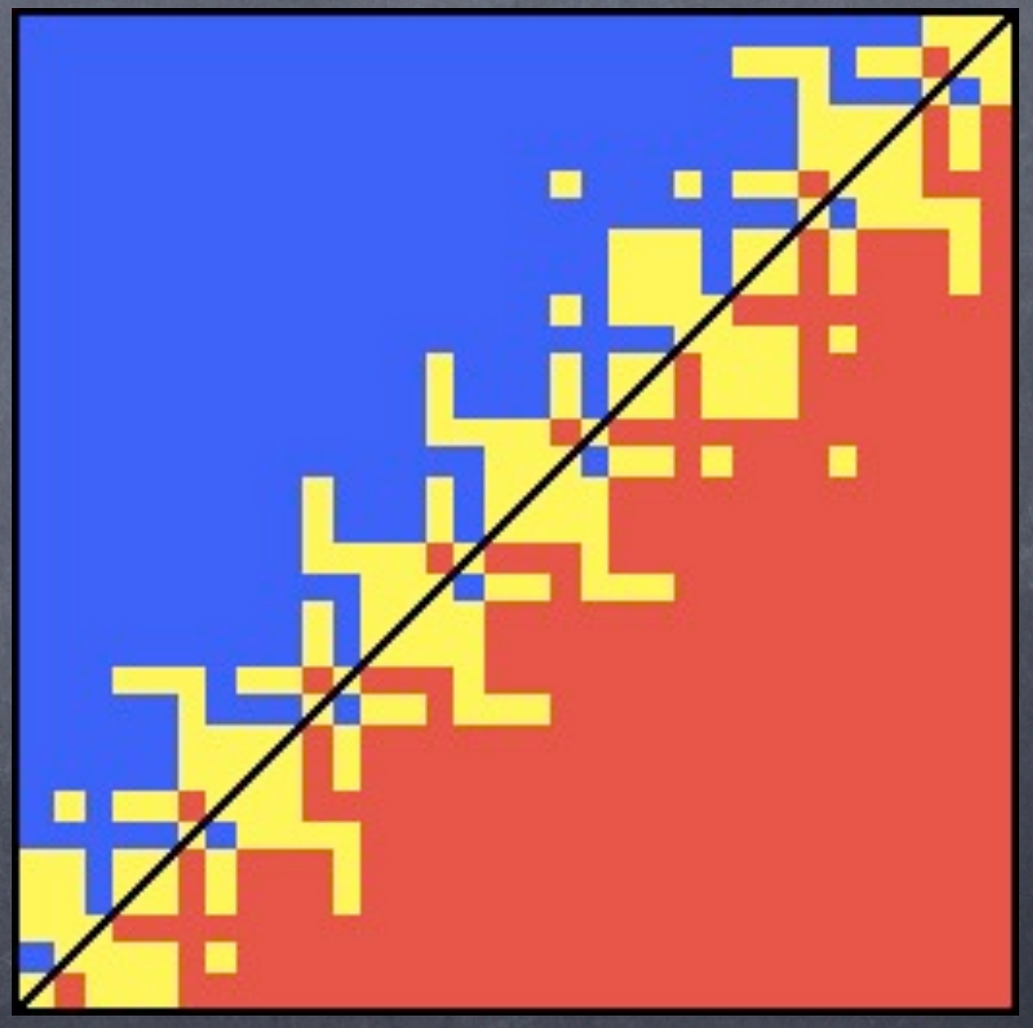
- q = (12, 12)
- r = (24, 24)
- p = (0.5, 0.5)
- O que esperaríamos ver?

1° Experimento

- q = (12, 12)
- r = (24, 24)
- p = (0.5, 0.5)
- O que esperaríamos ver?
- Faixa amarela bem definida

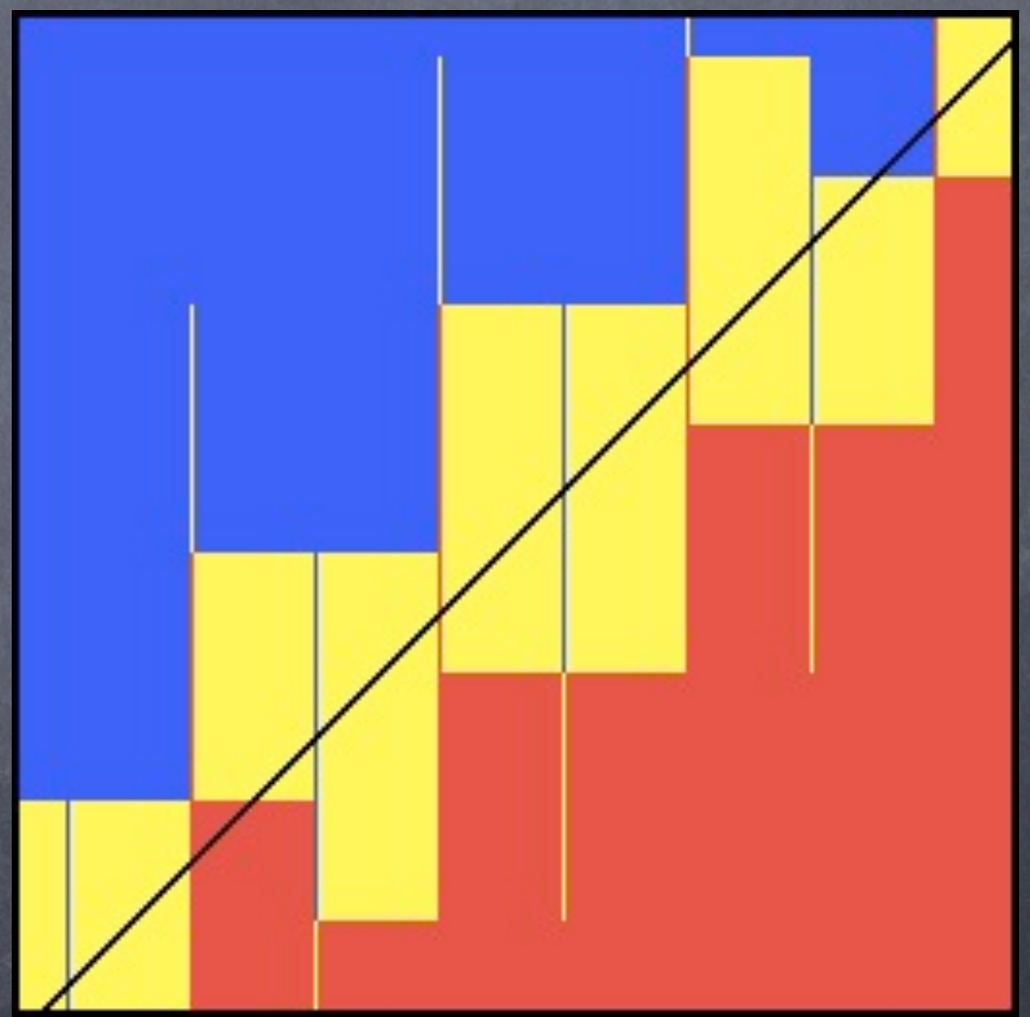
1° Experimento

- q = (12, 12)
- r = (24, 24)
- p = (0.5, 0.5)
- O que esperaríamos ver?
- Faixa amarela bem definida



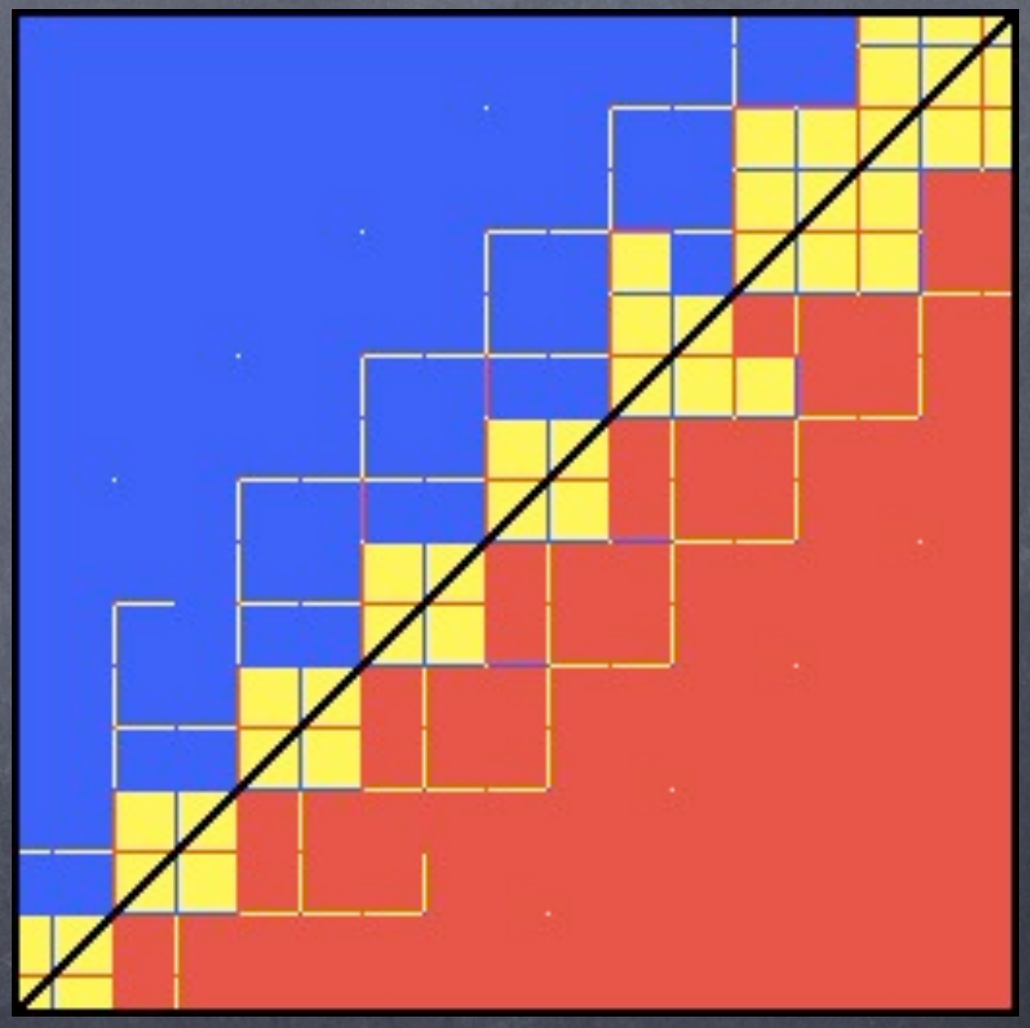
2° Experimento

- Parece uma função degrau
- Dividida por linhas azuis e vermelhas



3° Experimento

- Pontos isolados amarelos
- Blocos amarelos variando de tamanho
- Separados por linhas azuis e vermelhas



Escolha do Pivô

- Até agora usamos:
  $\text{sign}((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$

Escolha do Pivô

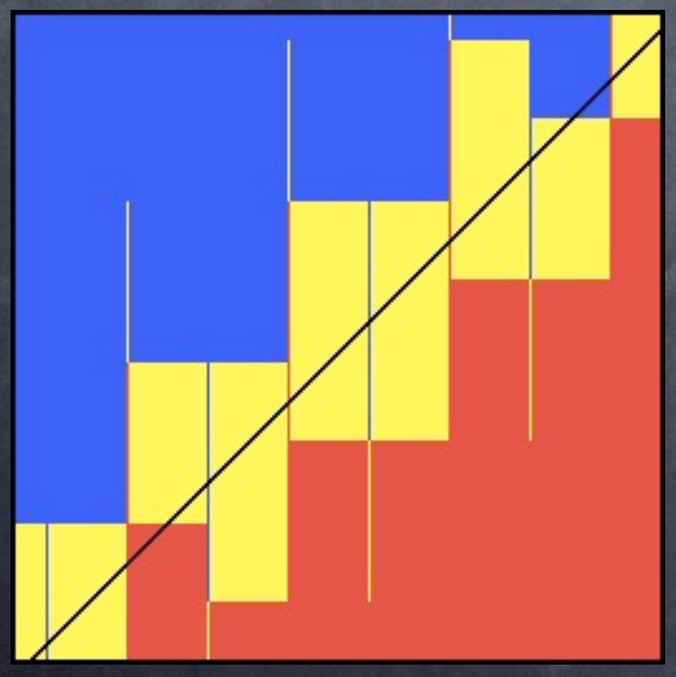
- Até agora usamos:
  $\text{sign}((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$

pivô: p

Escolha do Pivô

- Até agora usamos:
  $\text{sign}((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$

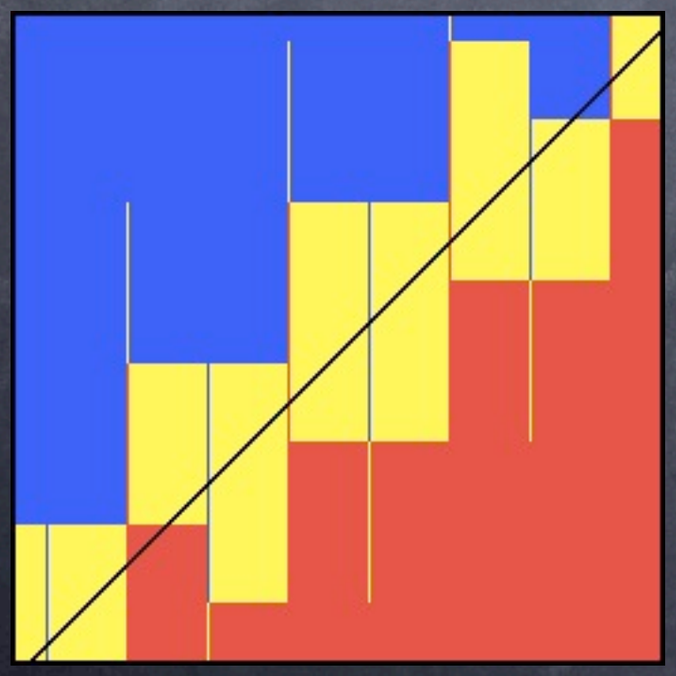
pivô: p



Escolha do Pivô

- Até agora usamos:
  $\text{sign}((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$

pivô: p

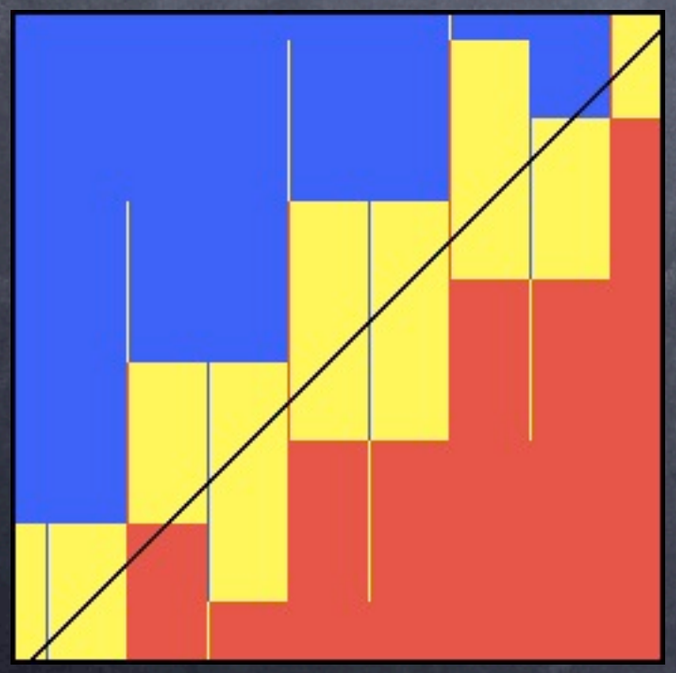


pivô: q

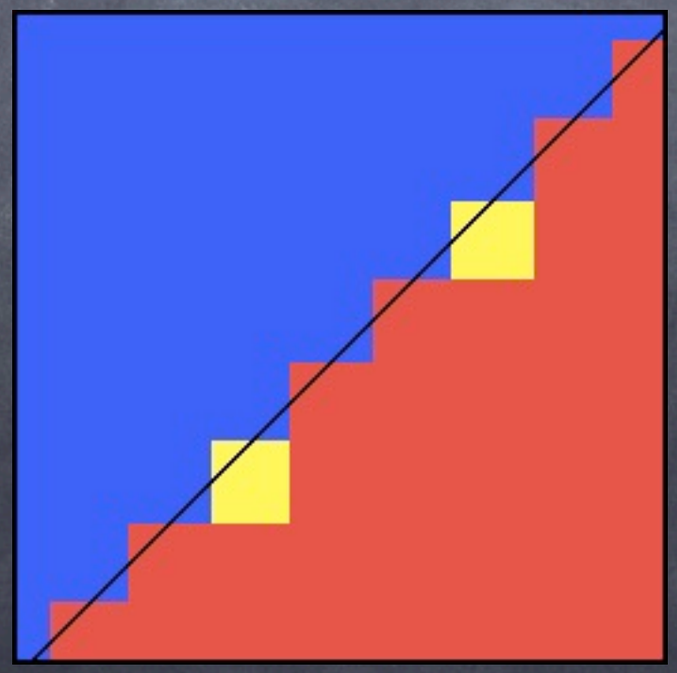
Escolha do Pivô

- Até agora usamos:
  $\text{sign}((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$

pivô: p



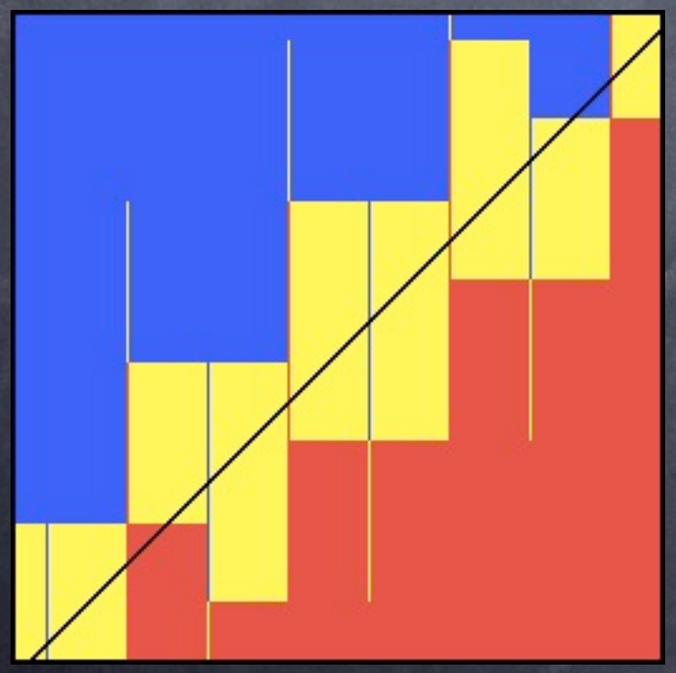
pivô: q



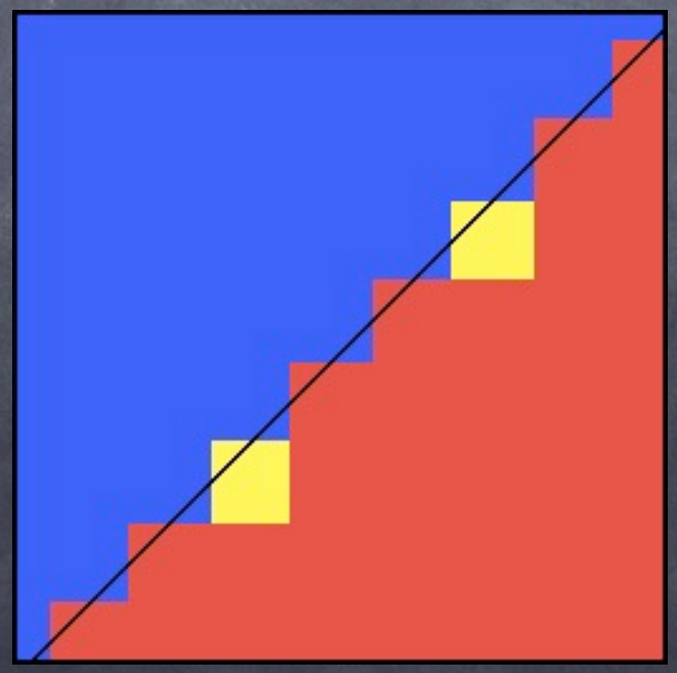
Escolha do Pivô

- Até agora usamos:
  $\text{sign}((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$

pivô: p



pivô: q

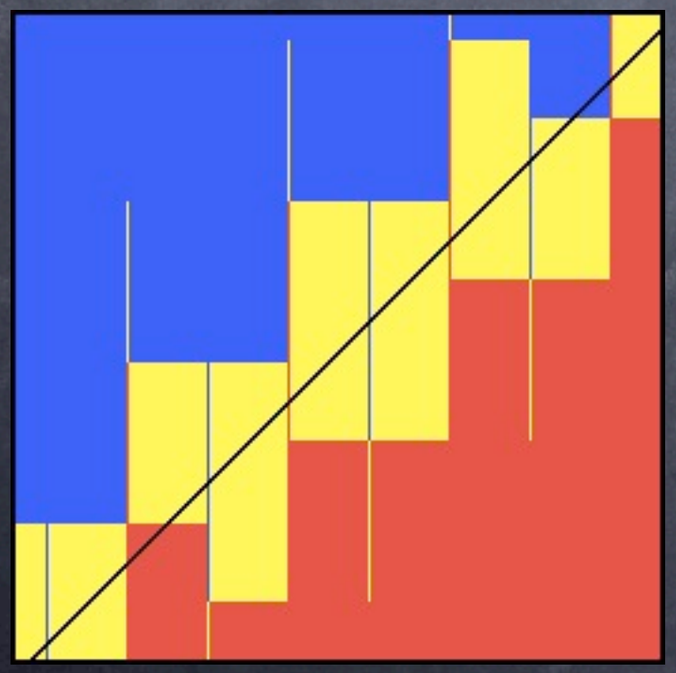


pivô: r

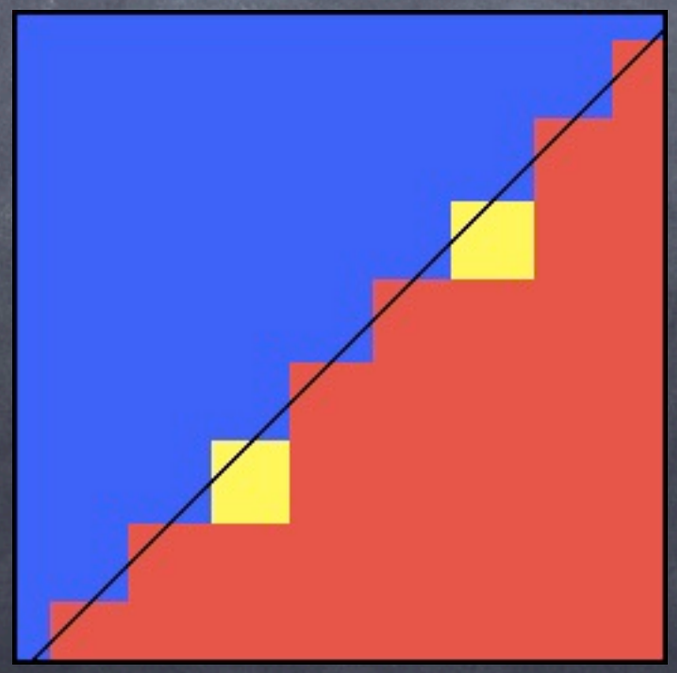
Escolha do Pivô

- Até agora usamos:
  $\text{sign}((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$

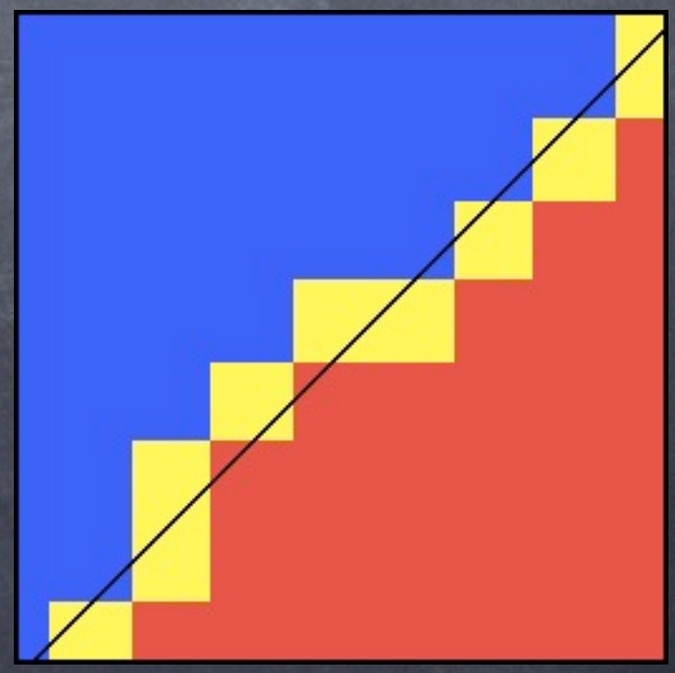
pivô: p



pivô: q



pivô: r



Fecho Convexo Incremental

Fecho Convexo Incremental

- Conjunto S de pontos.

Fecho Convexo Incremental

- Conjunto S de pontos.
- Dizemos que um vértice r enxerga uma aresta $e=(v_i,v_{i+1})$ se orientation$(v_i,v_{i+1},r) < 0$.

Fecho Convexo Incremental

- Conjunto S de pontos.
- Dizemos que um vértice r enxerga uma aresta $e=(v_i,v_{i+1})$ se orientation$(v_i,v_{i+1},r) < 0$.
- Se orientation$(v_i,v_{i+1},r) \leq 0$ então r enxerga fracamente e.

Duas propriedades

Duas propriedades

- A: Um ponto r está fora do fecho corrente se e somente existe i tal que r enxerga (vi,vi+1)

Duas propriedades

- A: Um ponto r está fora do fecho corrente se e somente existe i tal que r enxerga (vi,vi+1)
- B: O conjunto de arestas que r enxerga fracamente é uma subcadeia consecutiva.

Algoritmo Incremental

Algoritmo Incremental

1. Seja (a,b,c) um triângulo anti-horário

Algoritmo Incremental

1. Seja (a,b,c) um triângulo anti-horário
2. L <- (a,b,c)

Algoritmo Incremental

1. Seja (a,b,c) um triângulo anti-horário
2. L <- (a,b,c)
3. Remova a, b, c de S

Algoritmo Incremental

1. Seja (a,b,c) um triângulo anti-horário
2. L <- (a,b,c)
3. Remova a, b, c de S
4. para todo r em S faça

Algoritmo Incremental

1. Seja (a,b,c) um triângulo anti-horário
2. L <- (a,b,c)
3. Remova a, b, c de S
4. para todo r em S faça
5. se r enxerga uma aresta e então

Algoritmo Incremental

1. Seja (a,b,c) um triângulo anti-horário
2. L <- (a,b,c)
3. Remova a, b, c de S
4. para todo r em S faça
5. se r enxerga uma aresta e então
6. Compute (v_i,....,v_j) em L fracamente visíveis

Algoritmo Incremental

1. Seja (a,b,c) um triângulo anti-horário
2. L <- (a,b,c)
3. Remova a, b, c de S
4. para todo r em S faça
5. se r enxerga uma aresta e então
6. Compute (v_i,...,v_j) em L fracamente visíveis
7. Substitua (v_i,...,v_j) por r em L

Algoritmo Incremental

1. Seja (a,b,c) um triângulo anti-horário
2. L <- (a,b,c)
3. Remova a, b, c de S
4. para todo r em S faça
5. se r enxerga uma aresta e então
6. Compute (v_i,...,v_j) em L fracamente visíveis
7. Substitua (v_i,...,v_j) por r em L
8. devolva L

Subpassos

Subpassos

- Procuramos e percorrendo L, descartando r se e não existe

Subpassos

- Procuramos e percorrendo L, descartando r se e não existe
- Após achar e, andamos no sentido horário e anti-horário procurando v_i e v_j

Subpassos

- Procuramos e percorrendo L, descartando r se e não existe
- Após achar e, andamos no sentido horário e anti-horário procurando v_i e v_j
- Atualizamos L:
  offline: após achar v_i e v_j
  online: enquanto procuramos

Possíveis Falhas

Possíveis Falhas

- A_1: Um ponto fora não enxerga nenhuma aresta

Possíveis Falhas

- A_1: Um ponto fora não enxerga nenhuma aresta
- A_2: Um ponto dentro enxerga uma aresta

Possíveis Falhas

- A_1: Um ponto fora não enxerga nenhuma aresta
- A_2: Um ponto dentro enxerga uma aresta
- B_1: Um ponto fora enxerga todas as arestas

Possíveis Falhas

- A_1: Um ponto fora não enxerga nenhuma aresta
- A_2: Um ponto dentro enxerga uma aresta
- B_1: Um ponto fora enxerga todas as arestas
- B_2: Um ponto fora enxerga um conjunto não-contíguo de arestas

Primeira Falha

Primeira Falha

- A_1: Um ponto fora não enxerga nenhuma aresta

Primeira Falha

- A_1: Um ponto fora não enxerga nenhuma aresta

Pontos:

p_1 = (7.3000000000000194, (7.3000000000000167)

p_2 = (24.000000000000068, 24.000000000000071)

p_3 = (24.000000000000005, 24.000000000000053)

p_4 = (0.50000000000001621, (0.50000000000001243)

p_5 = (8,4) p_6 = (4,9) p_7 = (15,27)

p_8 = (26,25) p_9 = (19,11)

Primeira Falha

- A_1: Um ponto fora não enxerga nenhuma aresta

Pontos:

p_1 = (7.3000000000000194, (7.3000000000000167)

p_2 = (24.000000000000068, 24.000000000000071)

p_3 = (24.000000000000005, 24.000000000000053)

p_4 = (0.50000000000001621, (0.50000000000001243)

p_5 = (8,4) p_6 = (4,9) p_7 = (15,27)

p_8 = (26,25) p_9 = (19,11)

- Note que p_1, p_2, p_3 e p_4 são quase colineares.

Primeira Falha

- A_1: Um ponto fora não enxerga nenhuma aresta

Pontos:

p_1 = (7.3000000000000194, (7.3000000000000167)

p_2 = (24.000000000000068, 24.000000000000071)

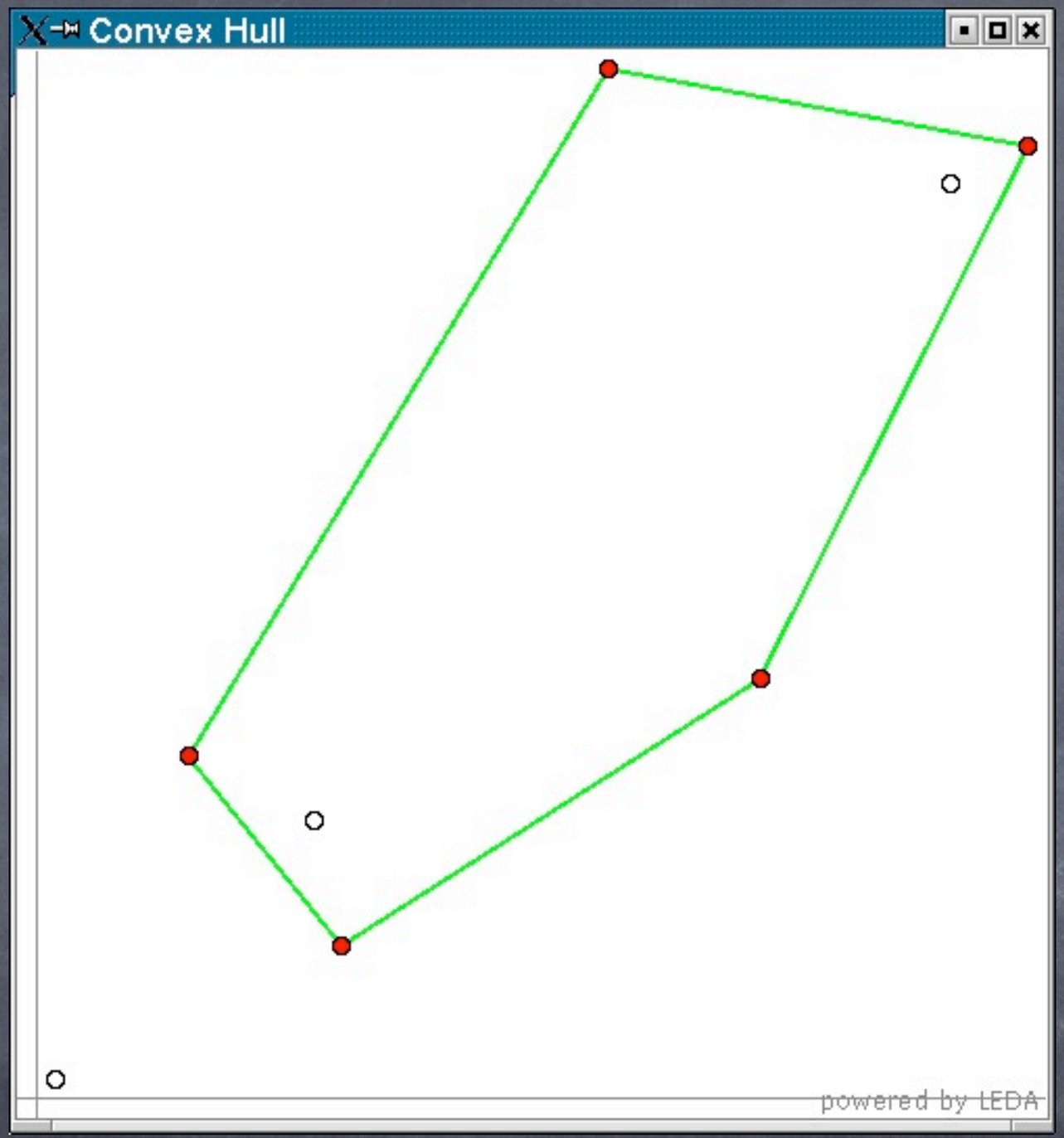
p_3 = (24.000000000000005, 24.000000000000053)

p_4 = (0.50000000000001621, (0.50000000000001243)

p_5 = (8,4) p_6 = (4,9) p_7 = (15,27)

p_8 = (26,25) p_9 = (19,11)

- Note que p_1, p_2, p_3 e p_4 são quase colineares.
- p_4 não enxerga nenhuma aresta



Primeiro Resultado

Segunda Falha

Segunda Falha

- A_2: Um ponto dentro enxerga uma aresta

Segunda Falha

- A_2: Um ponto dentro enxerga uma aresta
- p_4 está dentro do triângulo (p_1,p_2,p_3) mas enxerga a aresta (p_1,p_2)

Segunda Falha

- A_2: Um ponto dentro enxerga uma aresta
- p_4 está dentro do triângulo (p_1,p_2,p_3) mas enxerga a aresta (p_1,p_2)
- Resultado: Polígono (p_1,p_4,p_2,p_3) com um pouco de não-convexidade.

Terceira Falha

Terceira Falha

- B_1: Um ponto fora enxerga todas as arestas

Terceira Falha

- B_1: Um ponto fora enxerga todas as arestas
- p_4 enxerga todas as arestas

Terceira Falha

- B_1: Um ponto fora enxerga todas as arestas
- p_4 enxerga todas as arestas
- O que acontece? Depende:

Terceira Falha

- B_1: Um ponto fora enxerga todas as arestas
- p_4 enxerga todas as arestas
- O que acontece? Depende:
- off-line: nunca termina

Terceira Falha

- B_1: Um ponto fora enxerga todas as arestas
- p_4 enxerga todas as arestas
- O que acontece? Depende:
- off-line: nunca termina
- on-line: resposta errada ou quebra

Quarta Falha

Quarta Falha

- B_2: Um ponto fora enxerga um conjunto não-contíguo de arestas

Quarta Falha

- B_2: Um ponto fora enxerga um conjunto não-contíguo de arestas
- p_5 enxerga apenas (p_3,p_2), mas float_orient detecta que p_5 também enxerga (p_1,p_4).

Quarta Falha

- B_2: Um ponto fora enxerga um conjunto não-contíguo de arestas
- p_5 enxerga apenas (p_3,p_2), mas float_orient detecta que p_5 também enxerga (p_1,p_4).
- Como (p_1,p_4) vem primeiro na lista, o algoritmo insere p_5 sem remover vértices.

Quarta Falha

- B_2: Um ponto fora enxerga um conjunto não-contíguo de arestas
- p_5 enxerga apenas (p_3,p_2), mas float_orient detecta que p_5 também enxerga (p_1,p_4).
- Como (p_1,p_4) vem primeiro na lista, o algoritmo insere p_5 sem remover vértices.
- Temos um polígono com auto-interseção.

Efeitos Globais

Efeitos Globais

- Objetivo: eliminar a crença que o algoritmo sempre termina com uma aproximação do fecho convexo

Efeitos Globais

- Objetivo: eliminar a crença que o algoritmo sempre termina com uma aproximação do fecho convexo
- Encontra um fecho convexo que não inclui algum dos pontos (A_1)

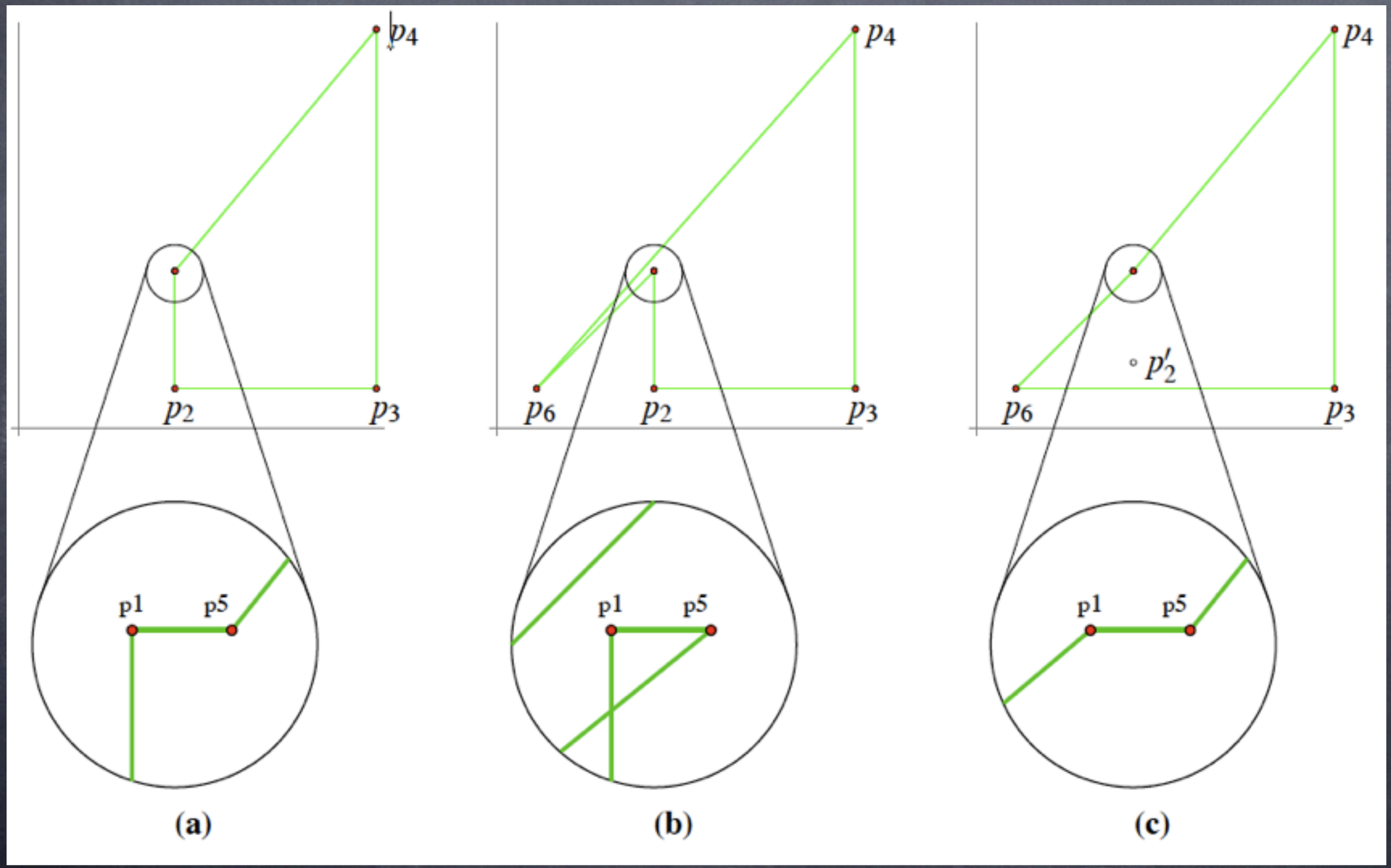
Efeitos Globais

- Objetivo: eliminar a crença que o algoritmo sempre termina com uma aproximação do fecho convexo
- Encontra um fecho convexo que não inclui algum dos pontos (A_1)
- O algoritmo não termina ou quebra (B_1)

Efeitos Globais

- Objetivo: eliminar a crença que o algoritmo sempre termina com uma aproximação do fecho convexo
- Encontra um fecho convexo que não inclui algum dos pontos (A_1)
- O algoritmo não termina ou quebra (B_1)
- O algoritmo encontra um polígono não-convexo.

Não-convexidade



Outros problemas

Outros problemas

- O paper aborda também:

Outros problemas

- O paper aborda também:
- Embrulho para Presente

Outros problemas

- O paper aborda também:
- Embrulho para Presente
- Triangulações de Delaunay 3D

Soluções

Soluções

- Garanta que orientation(p,q,r) devolva sempre o resultado correto: use racionais

Soluções

- Garanta que orientation(p,q,r) devolva sempre o resultado correto: use racionais
- Modifique o algoritmo para funcionar "bem" com float

Soluções

- Garanta que orientation(p,q,r) devolva sempre o resultado correto: use racionais
- Modifique o algoritmo para funcionar "bem" com float
- Faça uma perturbação da entrada para evitar casos problemáticos.

Bibliografia

Bibliografia

- Classroom Examples of Robustness Problems in Geometric Computations, L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, C. Yap, 2006

Bibliografia

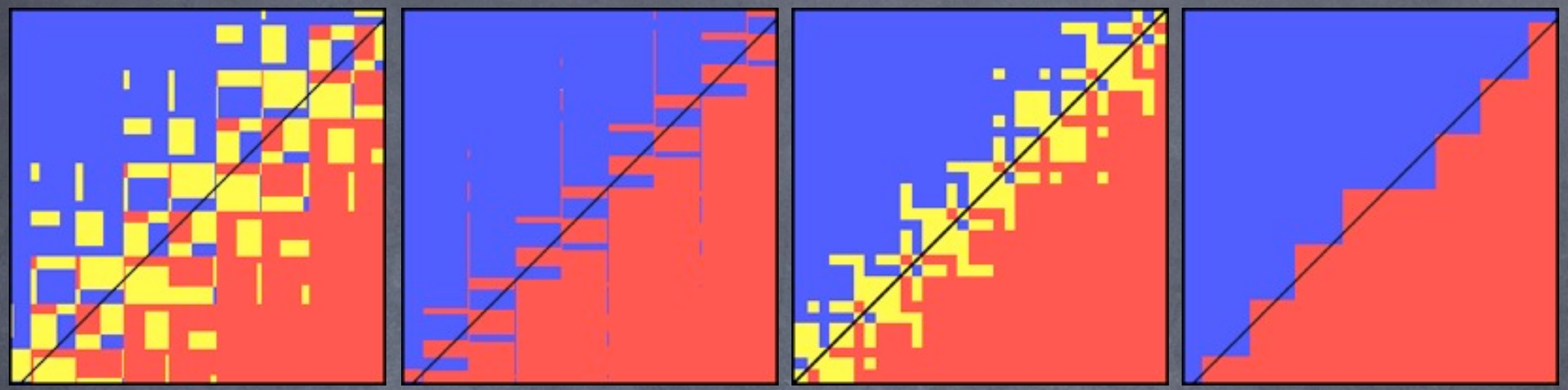
- Classroom Examples of Robustness Problems in Geometric Computations, L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, C. Yap, 2006
- Anatomy of Algorithmic Failures: A Case Study in Geometric Nonrobustness
  http://www.mpi-inf.mpg.de/~kettner/proj/NonRobust/

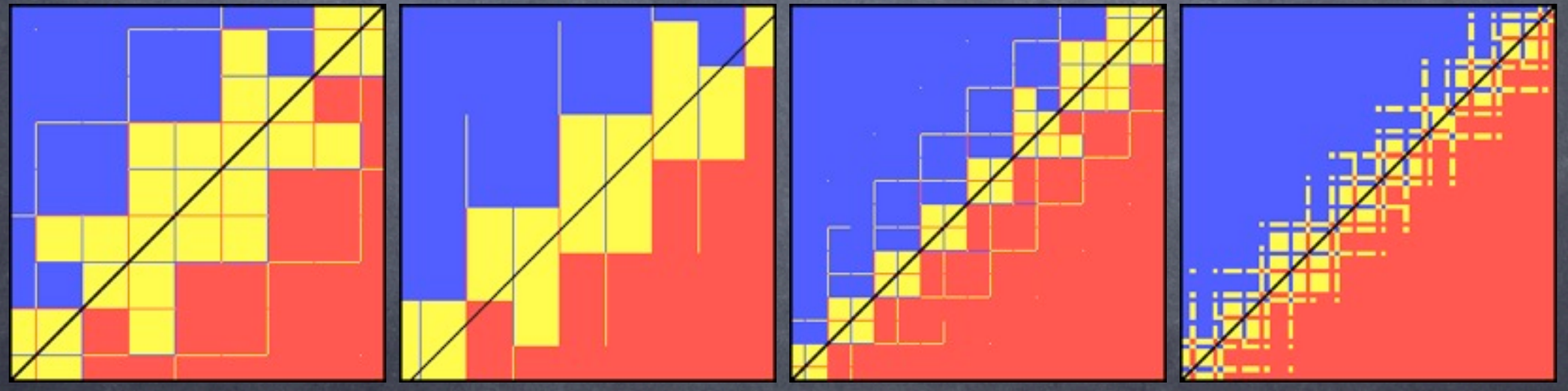
Bibliografia

- Classroom Examples of Robustness Problems in Geometric Computations, L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, C. Yap, 2006
- Anatomy of Algorithmic Failures: A Case Study in Geometric Nonrobustness
  http://www.mpi-inf.mpg.de/~kettner/proj/NonRobust/
- O site contém as imagens e códigos

Dúvidas?

Dúvidas?





Obrigado!