Fast oriented bounding box optimization on the rotation group $SO(3, \mathbb{R})$

Chia-Tche Chang[1], Bastien Gorissen[2,3] and Samuel Melchior[1,2]

chia-tche.chang@uclouvain.be
bastien.gorissen@cenaero.be
samuel.melchior@uclouvain.be

[1] Department of Mathematical Engineering (INMA), Université catholique de Louvain
[2] Department of Mechanical Engineering (MEMA), Université catholique de Louvain
[3] CENAERO

November 30th, 2012

Coming up next...

The problem

Given a set of n points $\mathcal{X}$ in 3D, find the minimum-volume arbitrarily oriented bounding box enclosing $\mathcal{X}$.

Collision detection, intersection tests, object representation, data approximation... (BV trees...)

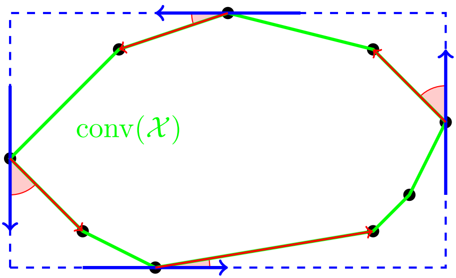
In 2D: the rotating calipers method

A minimum-area rectangle circumscribing a convex polygon has at least one side flush with an edge of the polygon.

In 2D: the rotating calipers method

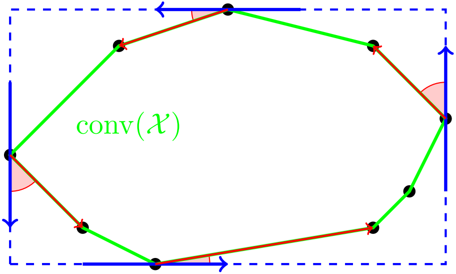
A minimum-area rectangle circumscribing a convex polygon has at least one side flush with an edge of the polygon.

Compute the convex hull: $\mathcal{O}(n \log n)$



In 2D: the rotating calipers method

A minimum-area rectangle circumscribing a convex polygon has at least one side flush with an edge of the polygon.



Compute the convex hull:
$\mathcal{O}(n \log n)$

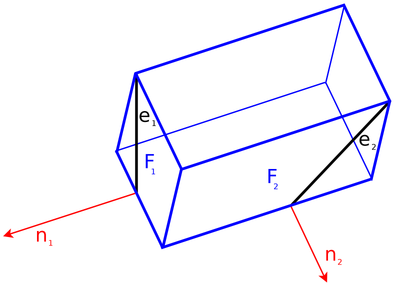
Loop on all edges:
$\mathcal{O}(n) \rightarrow$ easy and efficient

In 3D: generalization of the rotating calipers?

A minimum-volume box circumscribing a convex polyhedron has at least one face flush with a face of the polyhedron?

In 3D: generalization of the rotating calipers?

A minimum-volume box circumscribing a convex polyhedron has at least ~~one face~~ two adjacent faces flush with ~~a face~~ edges of the polyhedron. [O'Rourke, 1985]

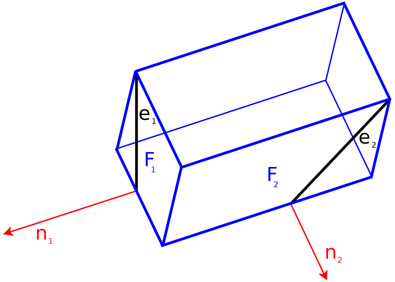


In 3D: generalization of the rotating calipers?

A minimum-volume box circumscribing a convex polyhedron has at least ~~one face~~ two adjacent faces flush with ~~a face~~ edges of the polyhedron. [O'Rourke, 1985]

Problem:

Loop on all pairs of edges and rotate the box while keeping edges flush → $\mathcal{O}(n^3)$ time complexity...



In practice...

O'Rourke's algorithm is too slow (cubic time)
$\rightarrow$ use faster but inexact methods:

In practice...

O'Rourke's algorithm is too slow (cubic time)
$\rightarrow$ use faster but inexact methods:

- PCA-based methods (covariance matrix):
  very fast and easy to compute but may be very inaccurate

In practice...

O'Rourke's algorithm is too slow (cubic time)
$\rightarrow$ use faster but inexact methods:

- PCA-based methods (covariance matrix):
  very fast and easy to compute but may be very inaccurate
- Brute-force all orientations with a small angle increment:
  large computation time and/or low accuracy

In practice...

O'Rourke's algorithm is too slow (cubic time)
$\rightarrow$ use faster but inexact methods:

- ◇ PCA-based methods (covariance matrix):
  very fast and easy to compute but may be very inaccurate
- ◇ Brute-force all orientations with a small angle increment:
  large computation time and/or low accuracy
- ◇ Brute-force a well-chosen set of orientations:
  may sometimes have (very) good accuracy but still too slow

In practice...

O'Rourke's algorithm is too slow (cubic time)
$\rightarrow$ use faster but inexact methods:

- PCA-based methods (covariance matrix):
  very fast and easy to compute but may be very inaccurate
- Brute-force all orientations with a small angle increment:
  large computation time and/or low accuracy
- Brute-force a well-chosen set of orientations:
  may sometimes have (very) good accuracy but still too slow
- Guaranteed quality approximation methods:
  same problem...

What do we want?

Goal:

- ◇ Very good accuracy: find an optimal OBB in (nearly?) all cases
- ◇ If a suboptimal solution is returned, it should be close to the best one
- ◇ Computational cost has to be low

What do we want?

Goal:

- ◇ Very good accuracy: find an optimal OBB in (nearly?) all cases
- ◇ If a suboptimal solution is returned, it should be close to the best one
- ◇ Computational cost has to be low

Our approach: iterative algorithm based on optimization methods

Coming up next...

A first, direct, formulation

$$\min_{\text{over size, position, orientation}} \quad \text{the volume of the bounding box}$$
$$\text{so that} \quad \text{all the points are in the box}$$

A first, direct, formulation

$$\min_{\text{over } \Delta\in\mathbb{R}^3,\ \text{position, orientation}} \quad \text{the volume of the bounding box}$$
$$\text{so that} \quad \text{all the points are in the box}$$

- $\Delta = (\Delta_\xi, \Delta_\eta, \Delta_\zeta)$ denotes the dimensions of the OBB,

A first, direct, formulation

$$\min_{\text{over } \Delta \in \mathbb{R}^3, \text{ position, orientation}} \quad \Delta_\xi \Delta_\eta \Delta_\zeta$$
$$\text{so that} \quad \text{all the points are in the box}$$

- $\Delta = (\Delta_\xi, \Delta_\eta, \Delta_\zeta)$ denotes the dimensions of the OBB,

A first, direct, formulation

$$\min_{\text{over } \Delta \in \mathbb{R}^3, \text{ position, orientation}} \Delta_\xi \Delta_\eta \Delta_\zeta$$

$$\text{so that} \quad -\tfrac{1}{2}\Delta \leq \begin{array}{c}\text{all the rotated and}\\ \text{centered points}\end{array} \leq \tfrac{1}{2}\Delta$$

- $\Delta = (\Delta_\xi, \Delta_\eta, \Delta_\zeta)$ denotes the dimensions of the OBB,

A first, direct, formulation

$$\min_{\text{over } \Delta\in\mathbb{R}^3, \Xi\in\mathbb{R}^3, \text{ orientation}} \quad \Delta_\xi \Delta_\eta \Delta_\zeta$$

$$\text{so that} \quad -\frac{1}{2}\Delta \leq \text{all the rotated points} - \Xi \leq \frac{1}{2}\Delta$$

- $\Delta = (\Delta_\xi, \Delta_\eta, \Delta_\zeta)$ denotes the dimensions of the OBB,
- Ξ is the center of the OBB.

A first, direct, formulation

$$\min_{\text{over } \Delta\in\mathbb{R}^3,\Xi\in\mathbb{R}^3,R\in SO(3,\mathbb{R})} \quad \Delta_\xi\Delta_\eta\Delta_\zeta$$
$$\text{so that} \quad -\tfrac{1}{2}\Delta \leq R(\text{all the points}) - \Xi \leq \tfrac{1}{2}\Delta$$

- $\Delta = (\Delta_\xi, \Delta_\eta, \Delta_\zeta)$ denotes the dimensions of the OBB,
- Ξ is the center of the OBB.
- $R \in SO(3, \mathbb{R})$ is a rotation matrix,

A first, direct, formulation

$$\min_{\text{over } \Delta\in\mathbb{R}^3, \Xi\in\mathbb{R}^3, R\in SO(3,\mathbb{R})} \quad \Delta_\xi \Delta_\eta \Delta_\zeta$$

$$\text{so that} \quad -\tfrac{1}{2}\Delta \leq R(\text{all the points}) - \Xi \leq \tfrac{1}{2}\Delta$$

- $\Delta = (\Delta_\xi, \Delta_\eta, \Delta_\zeta)$ denotes the dimensions of the OBB,
- Ξ is the center of the OBB.
- $R \in SO(3,\mathbb{R})$ is a rotation matrix,
- $SO(3,\mathbb{R}) = \left\{ R \in \mathbb{R}^{3\times 3} \mid R^T R = I = RR^T, \det(R) = 1 \right\},$

A first, direct, formulation

$$\begin{array}{ll} \min\limits_{\text{over } \Delta\in\mathbb{R}^3,\Xi\in\mathbb{R}^3,R\in SO(3,\mathbb{R})} & \Delta_\xi\Delta_\eta\Delta_\zeta \\ \text{s.t.} & -\frac{1}{2}\Delta \leq R\mathbf{X}_i - \Xi \leq \frac{1}{2}\Delta \quad \forall i = 1,\ldots,N \end{array}$$

- $\Delta = (\Delta_\xi, \Delta_\eta, \Delta_\zeta)$ denotes the dimensions of the OBB,
- Ξ is the center of the OBB.
- $R \in SO(3,\mathbb{R})$ is a rotation matrix,
- $SO(3,\mathbb{R}) = \left\{R \in \mathbb{R}^{3\times 3} \mid R^T R = I = RR^T, \det(R) = 1\right\}$,
- $\mathcal{X} = \{\mathbf{X}_i \mid i = 1,\ldots,N\}$ is the considered set of points

A first, direct, formulation

$$\begin{aligned} \min_{\Delta\in\mathbb{R}^3,\Xi\in\mathbb{R}^3,R\in SO(3,\mathbb{R})} \quad & \Delta_\xi\Delta_\eta\Delta_\zeta \\ \text{s.t.} \quad & -\tfrac{1}{2}\Delta \leq R\mathbf{X}_i - \Xi \leq \tfrac{1}{2}\Delta \quad \forall i = 1,\ldots,N \end{aligned}$$

- $\Delta = (\Delta_\xi, \Delta_\eta, \Delta_\zeta)$ denotes the dimensions of the OBB,
- Ξ is the center of the OBB.
- $R \in SO(3,\mathbb{R})$ is a rotation matrix,
- $SO(3,\mathbb{R}) = \left\{R \in \mathbb{R}^{3\times 3} \mid R^T R = I = RR^T, \det(R) = 1\right\}$,
- $\mathcal{X} = \{\mathbf{X}_i \mid i = 1,\ldots,N\}$ is the considered set of points

Smooth but constrained optimization problem

Unconstrained formulation

$$\min_{R\in SO(3,\mathbb{R})} \underbrace{\begin{pmatrix} \displaystyle\min_{\Delta\in\mathbb{R}^3,\Xi\in\mathbb{R}^3} & \Delta_\xi\Delta_\eta\Delta_\zeta \\ \text{s.t.} & -\frac{1}{2}\Delta \leq R\mathbf{X}_i - \Xi \leq \frac{1}{2}\Delta \ \forall i = 1,\ldots,N \end{pmatrix}}_{f(R)}$$

Unconstrained formulation

$$\min_{R\in SO(3,\mathbb{R})} \underbrace{\begin{pmatrix} \displaystyle\min_{\Delta\in\mathbb{R}^3,\Xi\in\mathbb{R}^3} & \Delta_\xi\Delta_\eta\Delta_\zeta \\ \text{s.t.} & -\frac{1}{2}\Delta \leq R\mathbf{X}_i - \Xi \leq \frac{1}{2}\Delta \ \forall i = 1,\dots,N \end{pmatrix}}_{f(R)}$$

The objective function $f(R)$ is simply the volume of the AABB of $\mathcal{X}$ rotated by R

Unconstrained formulation

$$\min_{R \in SO(3,\mathbb{R})} \underbrace{\begin{pmatrix} \displaystyle\min_{\Delta \in \mathbb{R}^3, \Xi \in \mathbb{R}^3} & \Delta_\xi \Delta_\eta \Delta_\zeta \\ \text{s.t.} & -\frac{1}{2}\Delta \leq R\mathbf{X}_i - \Xi \leq \frac{1}{2}\Delta \ \forall i = 1, \dots, N \end{pmatrix}}_{f(R)}$$

The objective function $f(R)$ is simply the volume of the AABB of $\mathcal{X}$ rotated by R

Unconstrained but non-differentiable optimization problem

Solving this problem requires...

Solving this problem requires...

- ... a derivative-free method

$f(R)$ is not differentiable everywhere...

Solving this problem requires...

- ... a derivative-free method
- ... a global search technique

$f(R)$ has many local minima...

Solving this problem requires...

- ... a derivative-free method
- ... a global search technique
- ... a fast convergence rate

That was the point!

Our idea: using an hybrid method

1. Use a global exploration component:
   genetic algorithm (GA)

Our idea: using an hybrid method

1. Use a global exploration component:
   genetic algorithm (GA)
2. Speed up convergence using a local exploitation algorithm
   Nelder-Mead simplex algorithm (NM)

Our idea: using an hybrid method

1. Use a global exploration component:
   genetic algorithm (GA)
2. Speed up convergence using a local exploitation algorithm
   Nelder-Mead simplex algorithm (NM)

- ◇ GA alone would be very slow to converge (GA more suitable for discrete search spaces)
- ◇ NM alone would be stuck in local minima (even with restarts)

Coming up next...

Global exploration: genetic algorithms

Stochastic population-based evolutionary method

(original variant proposed by Holland in the 1970s)

- ◇ Population-based: keep a large set of candidates at each iteration
- ◇ Evolutionary: generate new candidates by combining current ones depending on their performance

The general framework

Start with a set of candidates (*population*)
and a performance function (*fitness function*)

The general framework

Start with a set of candidates (*population*)
and a performance function (*fitness function*)

At each *generation*:

- Selection: *parents* are selected depending on their fitness

The general framework

Start with a set of candidates (*population*)
and a performance function (*fitness function*)

At each *generation*:

- ◇ Selection: *parents* are selected depending on their fitness
- ◇ Crossover: selected parents produce *offsprings*

The general framework

Start with a set of candidates (*population*)
and a performance function (*fitness function*)

At each *generation*:

- Selection: *parents* are selected depending on their fitness
- Crossover: selected parents produce *offsprings*
- Mutation: offsprings can be subject to *mutations*
  (random modification, gradient step, SA step, ...)

The Nelder-Mead simplex algorithm

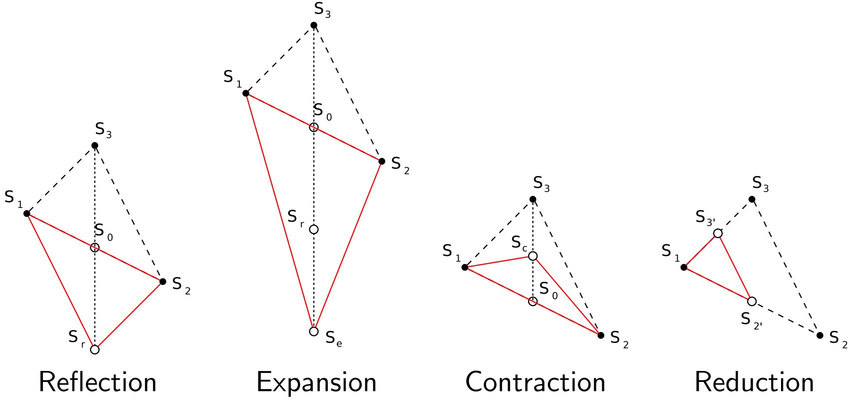
Derivative-free simplicial optimization method

(original algorithm proposed by Nelder & Mead in 1965)

Simplex (in $\mathbb{R}^n$) $=$ set of $n+1$ affinely independent points

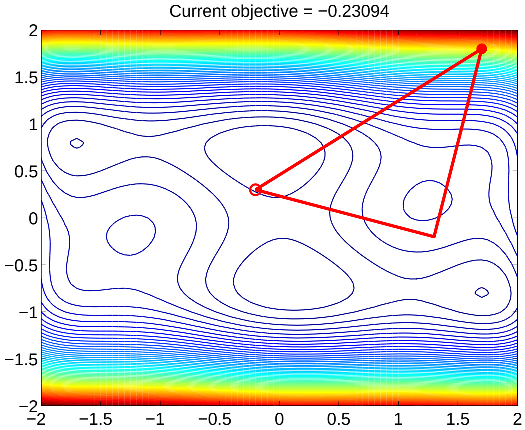
- $n = 2$: triangle
- $n = 3$: tetrahedron
- ...

Ideas of the algorithm (details omitted)



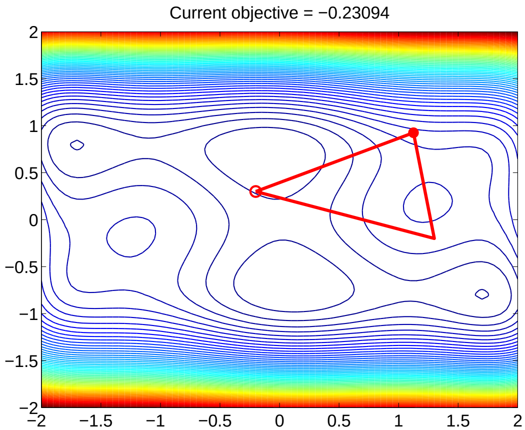
Four main ways to move/transform the simplex depending on the performance of its vertices: affine combinations

Nelder-Mead and the six-hump camel back...



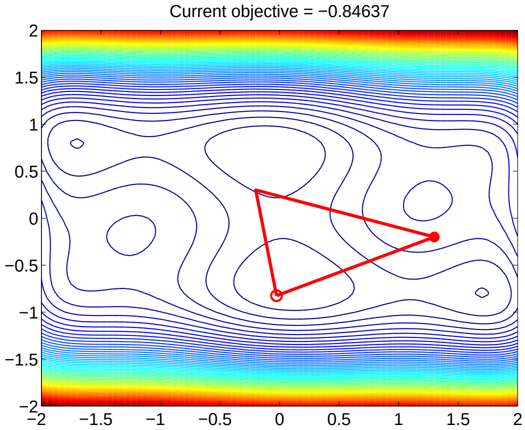
The initial simplex

Nelder-Mead and the six-hump camel back...



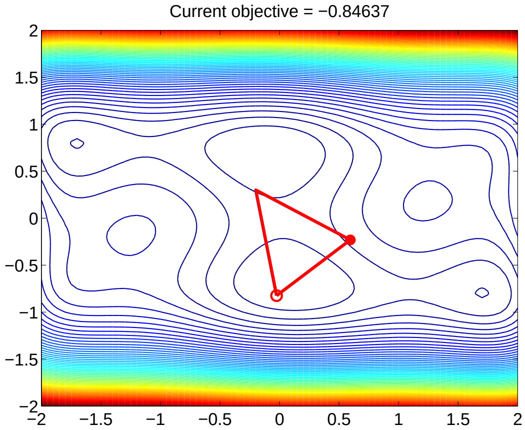
Iteration 1: contraction

Nelder-Mead and the six-hump camel back...



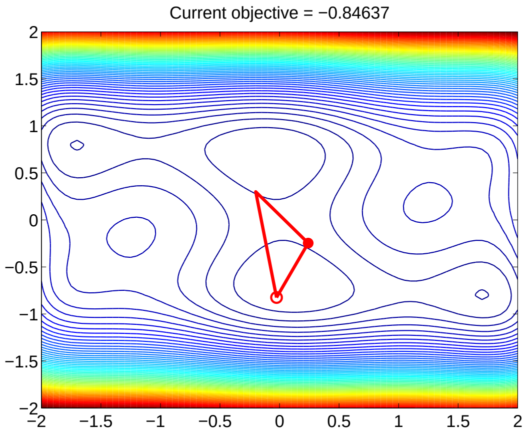
Iteration 2: reflection

Nelder-Mead and the six-hump camel back...



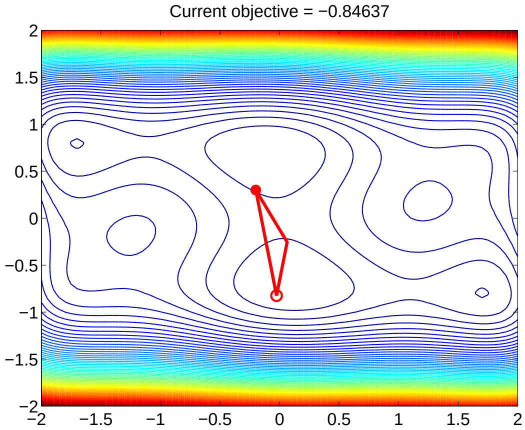
Iteration 3: contraction

Nelder-Mead and the six-hump camel back...



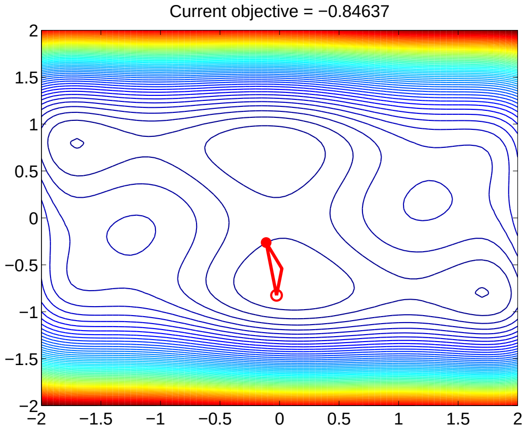
Iteration 4: contraction

Nelder-Mead and the six-hump camel back...



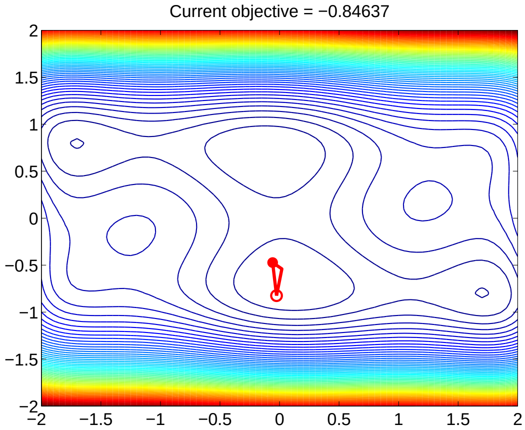
Iteration 5: contraction

Nelder-Mead and the six-hump camel back...



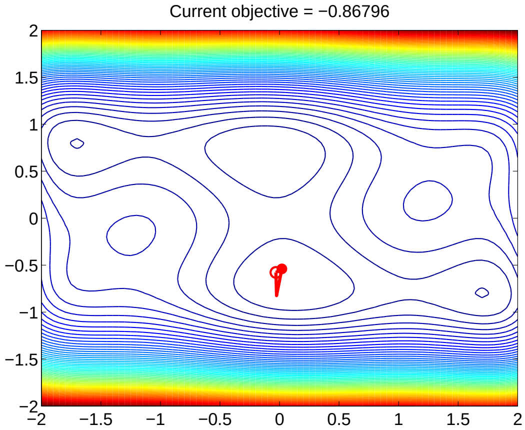
Iteration 6: reduction

Nelder-Mead and the six-hump camel back...



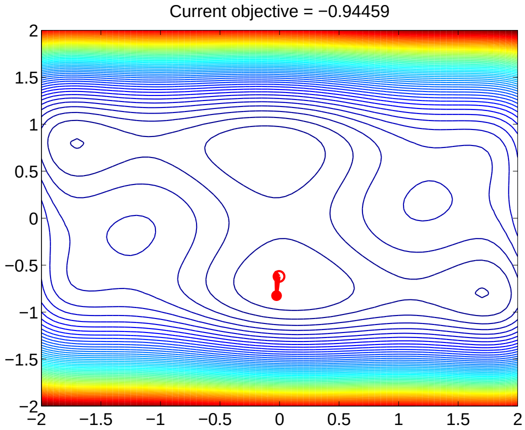
Iteration 7: contraction

Nelder-Mead and the six-hump camel back...



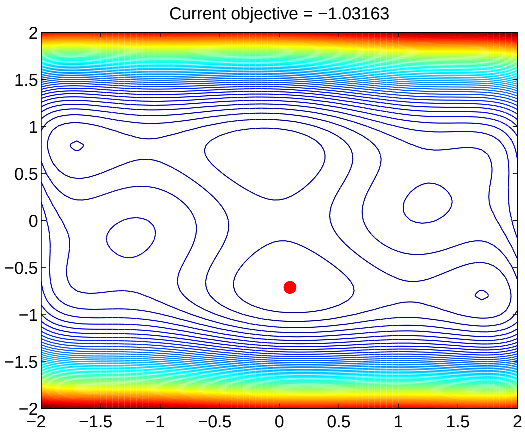
Some more iterations...

Nelder-Mead and the six-hump camel back...



Some more iterations...

Nelder-Mead and the six-hump camel back...



The final result

Mixing NM and GA together (simplified version)

Population of M simplices (simplex = set of 4 rotation matrices)
Fitness function $f(R)$ (volume of corresponding OBB)

Mixing NM and GA together (simplified version)

Population of M simplices (simplex = set of 4 rotation matrices)
Fitness function $f(R)$ (volume of corresponding OBB)

- ◇ Selection: Evaluate fitness of all simplices, keep best 50%

Mixing NM and GA together (simplified version)

Population of M simplices (simplex = set of 4 rotation matrices)
Fitness function $f(R)$ (volume of corresponding OBB)

- ◇ Selection: Evaluate fitness of all simplices, keep best 50%
- ◇ Crossover I: Create $\frac{M}{2}$ offsprings by mixing vertices:

$$A_1 B_1 C_1 D_1 \otimes A_2 B_2 C_2 D_2 \to A_{i_1} B_{i_2} C_{i_3} D_{i_4}, \quad i_k \in \{1, 2\}$$

Mixing NM and GA together (simplified version)

Population of M simplices (simplex = set of 4 rotation matrices)
Fitness function $f(R)$ (volume of corresponding OBB)

- Selection: Evaluate fitness of all simplices, keep best 50%
- Crossover I: Create $\frac{M}{2}$ offsprings by mixing vertices:
  $$A_1B_1C_1D_1 \otimes A_2B_2C_2D_2 \to A_{i_1}B_{i_2}C_{i_3}D_{i_4}, \quad i_k \in \{1,2\}$$
- Crossover II: Create $\frac{M}{2}$ offsprings by affinely combine vertices:
  $A_1B_1C_1D_1 \otimes A_2B_2C_2D_2 \to A_3B_3C_3D_3$ with $A_3 = \lambda A_1 + (1-\lambda)A_2$

Mixing NM and GA together (simplified version)

Population of M simplices (simplex $=$ set of 4 rotation matrices)
Fitness function $f(R)$ (volume of corresponding OBB)

- Selection: Evaluate fitness of all simplices, keep best 50%
- Crossover I: Create $\frac{M}{2}$ offsprings by mixing vertices:
$$A_1 B_1 C_1 D_1 \otimes A_2 B_2 C_2 D_2 \rightarrow A_{i_1} B_{i_2} C_{i_3} D_{i_4}, \quad i_k \in \{1, 2\}$$
- Crossover II: Create $\frac{M}{2}$ offsprings by affinely combine vertices:
$A_1 B_1 C_1 D_1 \otimes A_2 B_2 C_2 D_2 \rightarrow A_3 B_3 C_3 D_3$ with $A_3 = \lambda A_1 + (1 - \lambda) A_2$
- Mutation: Apply K Nelder-Mead iterations on each offspring

HYBBRID

Nelder-Mead algorithm $\oplus$ Genetic algorithm
on the special orthogonal group $SO(3)$
to solve the optimal OBB problem

HYBBRID

Nelder-Mead algorithm $\oplus$ Genetic algorithm
on the special orthogonal group $SO(3)$
to solve the optimal OBB problem

$$=$$

HYbrid Bounding Box Rotation IDentification algorithm

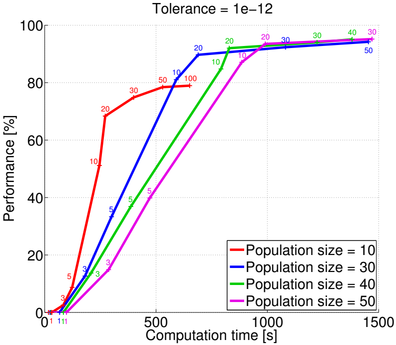
Coming up next...

Behavior of HYBBRID

All algorithms tested on a benchmark set of ~ 300 objects (Gamma db)
Implementations done in MATLAB (built-in functions are used)

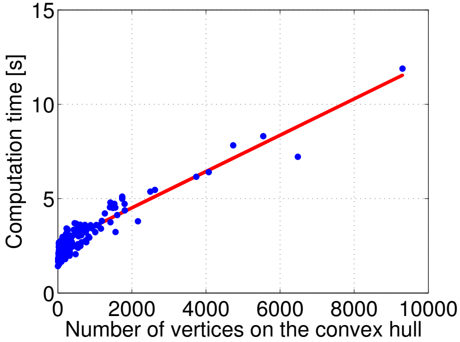
Behavior of HYBBRID: yes, it works!

All algorithms tested on a benchmark set of ~ 300 objects (Gamma db)
Implementations done in MATLAB (built-in functions are used)



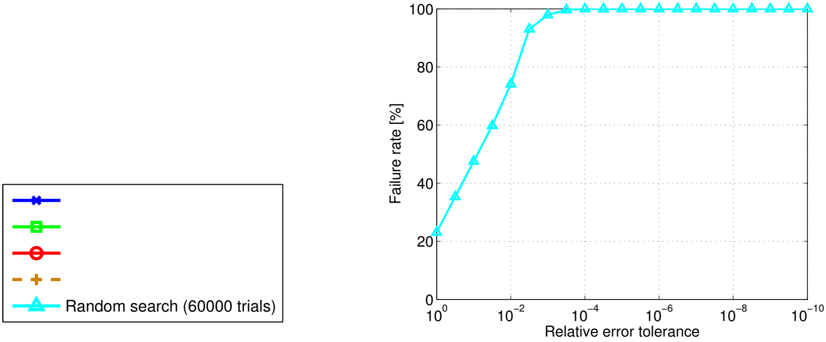
Error is less than 10^{-12} in $90\%+$ of the cases!

How does it scale?



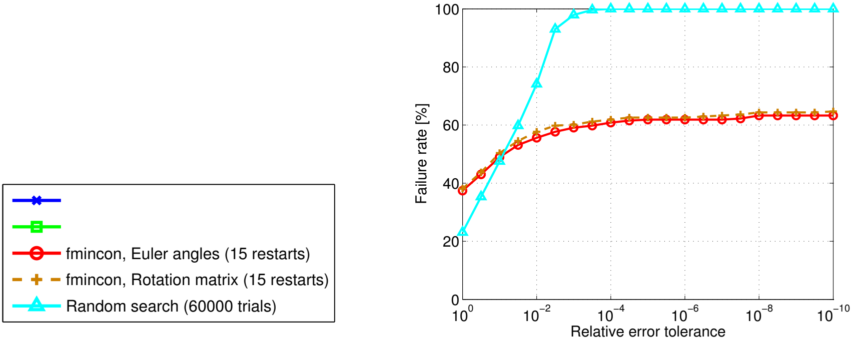
Experimental results show a roughly linear complexity!

Comparison to other iterative approaches



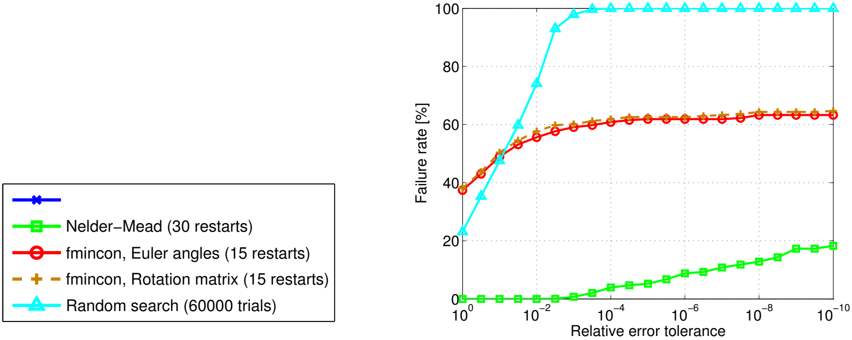
Trying random orientations does not work...

Comparison to other iterative approaches



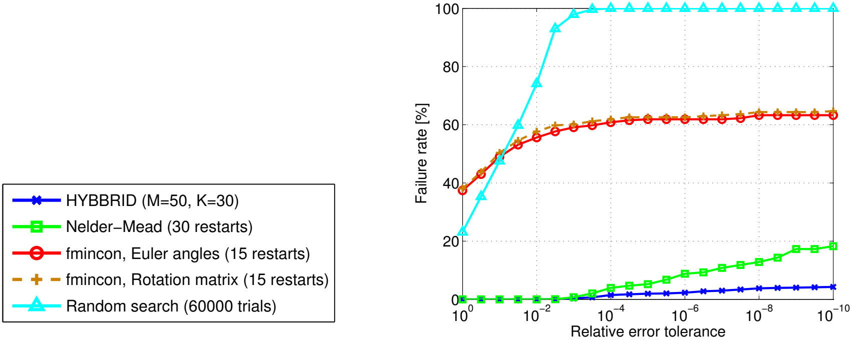
Constrained smooth opti: success rate $\sim 40\%$ but mainly AABBs...

Comparison to other iterative approaches



Unconstrained non-diff. opti, random initializations: much better results!

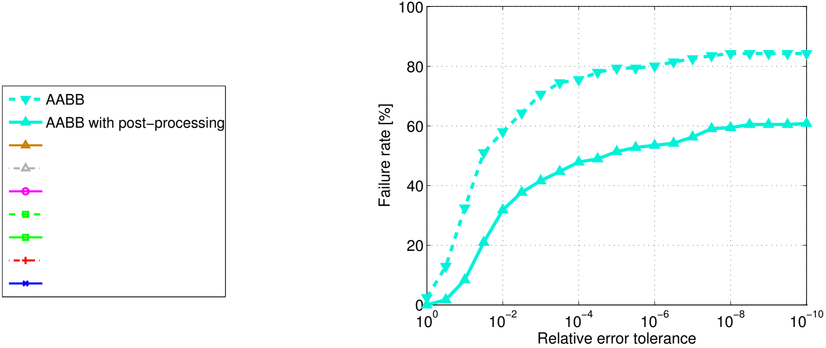
Comparison to other iterative approaches



HYBBRID: combining potential solutions does improve the success rate!

Comparison to other algorithms

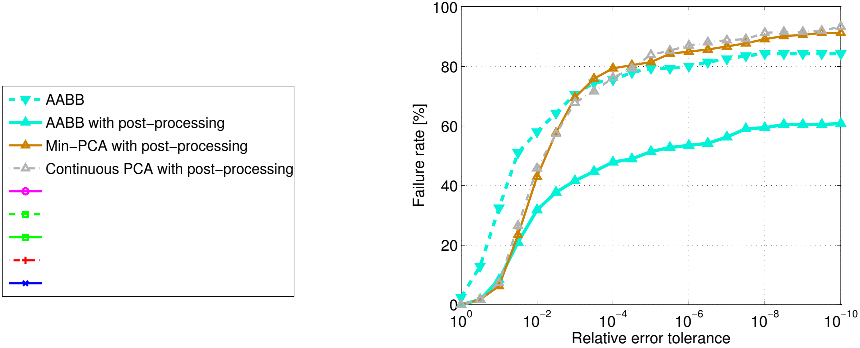
First, let's ignore the computational cost and look at the failure rates...



A reference point: the simple AABB

Comparison to other algorithms

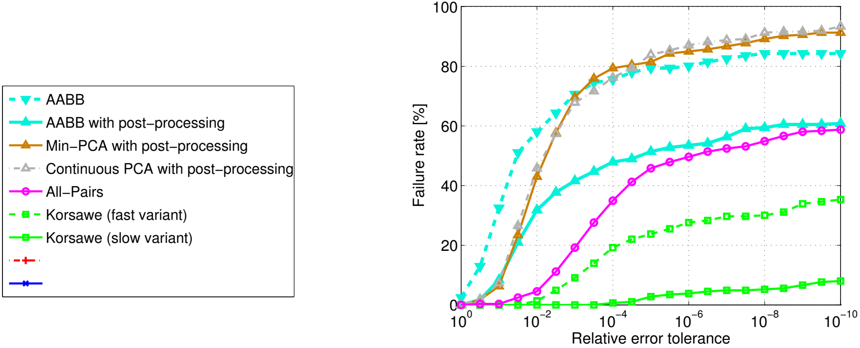
First, let's ignore the computational cost and look at the failure rates...



PCA-based methods: limited accuracy

Comparison to other algorithms

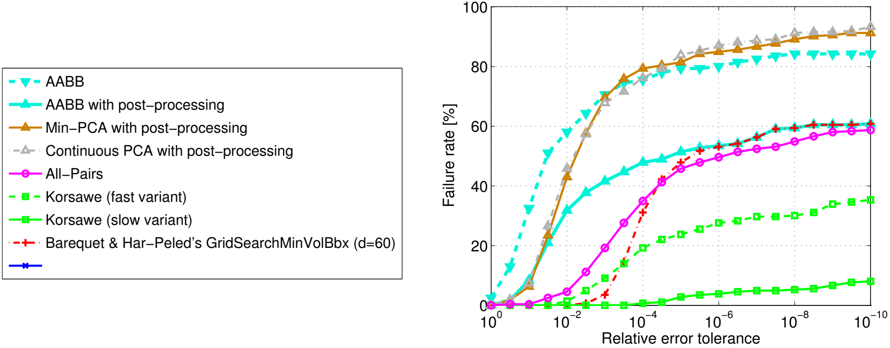
First, let's ignore the computational cost and look at the failure rates...



Brute-forcing on a set of orientations *may* be OK... if well chosen!

Comparison to other algorithms

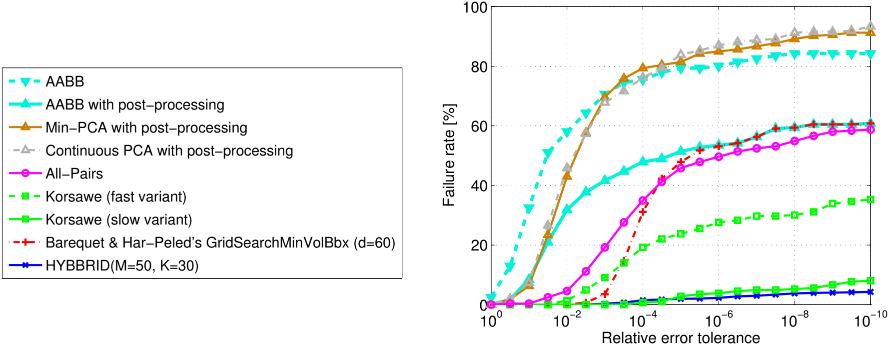
First, let's ignore the computational cost and look at the failure rates...



Guaranteed approximation algorithms: limited by computational resources

Comparison to other algorithms

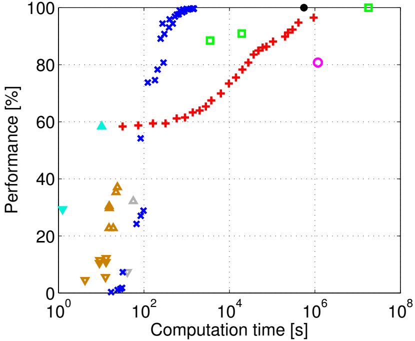
First, let's ignore the computational cost and look at the failure rates...

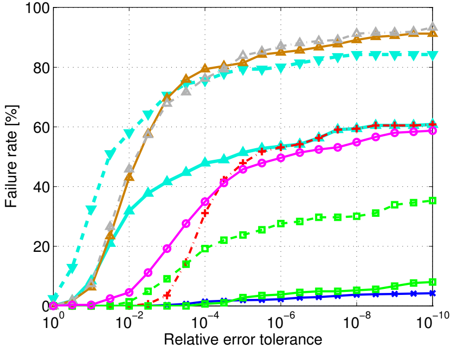


HYBBRID: more accurate than these other methods

Comparison to other algorithms

What are the computation times? (Tolerance: 10^{-3})

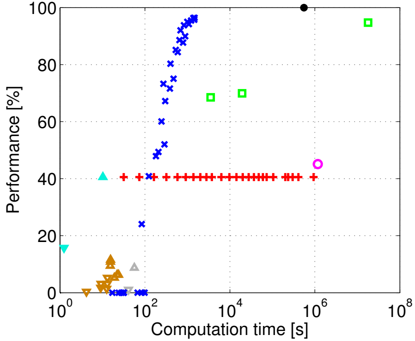


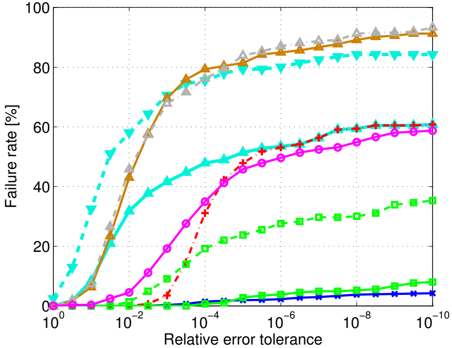


AABB – PCA – Continuous PCA – Brute-force on a set of orientations
O'Rourke's exact algorithm – Guaranteed approximation – HYBBRID

Comparison to other algorithms

What are the computation times? (Tolerance: 10^{-6})





AABB – PCA – Continuous PCA – Brute-force on a set of orientations
O'Rourke's exact algorithm – Guaranteed approximation – HYBBRID

Coming up next...

Conclusion

- HYBBRID: Nelder-Mead $\oplus$ Genetic algorithm
  able to approximate optimal OBBs using optimization on $SO(3)$

Conclusion

- HYBBRID: Nelder-Mead $\oplus$ Genetic algorithm
  able to approximate optimal OBBs using optimization on $SO(3)$
- More accurate and/or faster than other algorithms

Conclusion

- HYBBRID: Nelder-Mead $\oplus$ Genetic algorithm
  able to approximate optimal OBBs using optimization on $SO(3)$
- More accurate and/or faster than other algorithms
- Still has room for improvements...

Conclusion

- HYBBRID: Nelder-Mead $\oplus$ Genetic algorithm
  able to approximate optimal OBBs using optimization on $SO(3)$
- More accurate and/or faster than other algorithms
- Still has room for improvements...

Thank you for your attention!