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
The problem: minimum-volume OBB

Coming up next...

The problem: minimum-volume OBB
Exact methods in 2D and 3D
Classical approaches for the 3D case
Our goal

Bringing optimization into the game

How to solve an optimization problem?

Results

Conclusion
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: $O(n \log n)$
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:
$O(n \log n)$

Loop on all edges:
$O(n) \rightarrow$ 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]
The problem: minimum-volume OBB
Exact methods in 2D and 3D

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 → $O(n^3)$ time complexity...
In practice...

O’Rourke’s algorithm is too slow (cubic time)

→ use faster but inexact methods:
In practice...

O’Rourke’s algorithm is too slow (cubic time)

→ use faster but inexact methods:

◊ PCA-based methods (covariance matrix):
  very fast and easy to compute but may be very inaccurate
The problem: minimum-volume OBB

Classical approaches for the 3D case

In practice...

O’Rourke’s algorithm is too slow (cubic time)
→ 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)
→ 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)
→ 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...

The problem: minimum-volume OBB

Bringing optimization into the game
- OBB fitting as an optimization problem
- Requirements
- Going hybrid

How to solve an optimization problem?

Results

Conclusion
A first, direct, formulation

\[
\min \quad \text{over size, position, orientation} \\
\text{so that} \\
\quad \text{the volume of the bounding box} \\
\quad \text{all the points are in the box}
\]
A first, direct, formulation

\[
\min_{\Delta \in \mathbb{R}^3, \text{position, orientation}} \quad \text{the volume of the bounding box}
\]

so that

\[
\text{all the points are in the box}
\]

\[
\Delta = (\Delta_\xi, \Delta_\eta, \Delta_\zeta) \text{ denotes the dimensions of the OBB,}
\]
A first, direct, formulation

\[
\begin{align*}
\min_{\Delta \in \mathbb{R}^3, \text{ position, orientation}} \quad & \Delta_{\xi} \Delta_{\eta} \Delta_{\zeta} \\
\text{so that} \quad & \text{all the points are in the box} \\
\diamond \Delta = (\Delta_{\xi}, \Delta_{\eta}, \Delta_{\zeta}) \text{ denotes the dimensions of the OBB,}
\end{align*}
\]
A first, direct, formulation

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

so that

$$-\frac{1}{2} \Delta \leq \text{all the rotated and centered points} \leq \frac{1}{2} \Delta$$

$\Diamond \Delta = (\Delta_\xi, \Delta_\eta, \Delta_\zeta)$ denotes the dimensions of the OBB,
A first, direct, formulation

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

so that

\[-\frac{1}{2} \Delta \leq \text{all the rotated points} - \Xi \leq \frac{1}{2} \Delta\]

- $\Delta = (\Delta_\xi, \Delta_\eta, \Delta_\zeta)$ denotes the dimensions of the OBB,
- $\Xi$ is the center of the OBB.
A first, direct, formulation

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

- \( \Delta = (\Delta \xi, \Delta \eta, \Delta \zeta) \) denotes the dimensions of the OBB,
- \( \Xi \) is the center of the OBB,
- \( R \in SO(3, \mathbb{R}) \) is a rotation matrix,
A first, direct, formulation

\[
\begin{align*}
\min_{\Delta \in \mathbb{R}^3, \Xi \in \mathbb{R}^3, R \in SO(3, \mathbb{R})} & \quad \Delta \xi \Delta \eta \Delta \zeta \\
& \quad - \frac{1}{2} \Delta \leq R(\text{all the points}) - \Xi \leq \frac{1}{2} \Delta 
\end{align*}
\]

- \(\Delta \) denotes the dimensions of the OBB,
- \(\Xi\) is the center of the OBB.
- \(R \in SO(3, \mathbb{R})\) is a rotation matrix.
- \(SO(3, \mathbb{R}) = \{ R \in \mathbb{R}^{3\times3} \mid R^T R = I = RR^T, \det(R) = 1 \}\).
Bringing optimization into the game

OBB fitting as an optimization problem

A first, direct, formulation

\[
\begin{align*}
\min_{\Delta \in \mathbb{R}^3, \Xi \in \mathbb{R}^3, R \in SO(3, \mathbb{R})} & \quad \Delta_x \Delta_y \Delta_z \\
\text{s.t.} & \quad -\frac{1}{2} \Delta \leq R X_i - \Xi \leq \frac{1}{2} \Delta \quad \forall i = 1, \ldots, N
\end{align*}
\]

- $\Delta = (\Delta_x, \Delta_y, \Delta_z)$ denotes the dimensions of the OBB,
- $\Xi$ is the center of the OBB.
- $R \in SO(3, \mathbb{R})$ is a rotation matrix,
- $SO(3, \mathbb{R}) = \{ R \in \mathbb{R}^{3 \times 3} \mid R^T R = I = RR^T, \det(R) = 1 \}$,
- $\mathcal{X} = \{ X_i \mid i = 1, \ldots, N \}$ is the considered set of points
A first, direct, formulation

\[
\begin{align*}
\min_{\Delta \in \mathbb{R}^3, \Xi \in \mathbb{R}^3, R \in SO(3, \mathbb{R})} & \quad \Delta_\xi \Delta_\eta \Delta_\zeta \\
\text{s.t.} & \quad -\frac{1}{2} \Delta \leq R X_i - \Xi \leq \frac{1}{2} \Delta \quad \forall i = 1, \ldots, N
\end{align*}
\]

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

Smooth but constrained optimization problem
Unconstrained formulation

\[
\min_{R \in SO(3, \mathbb{R})} \left( \min_{\Delta \in \mathbb{R}^3, \Xi \in \mathbb{R}^3} \begin{array}{c}
\Delta_x \\
\frac{1}{2} \Delta \leq RX_i - \Xi \leq \frac{1}{2} \Delta \forall i = 1, \ldots, N
\end{array} \right)
\]

The objective function \( f(R) \) is simply the volume of the AABB of \( X \) rotated by \( R \).
Unconstrained formulation

\[
\min_{R \in SO(3, \mathbb{R})} \left( \begin{array}{c}
\min_{\Delta \in \mathbb{R}^3, \Xi \in \mathbb{R}^3} \Delta_x \Delta_y \Delta_z \\
-\frac{1}{2} \Delta \leq RX_i - \Xi \leq \frac{1}{2} \Delta \quad \forall i = 1, \ldots, N
\end{array} \right)
\]

\[f(R)\]

The objective function \(f(R)\) is simply the volume of the AABB of \(X\) rotated by \(R\).
Unconstrained formulation

\[
\min_{R \in SO(3, \mathbb{R})} \left( \min_{\Delta \in \mathbb{R}^3, \Xi \in \mathbb{R}^3} \begin{array}{c}
\Delta_x \Delta_y \Delta_z \\
-\frac{1}{2} \Delta \leq R\mathbf{X}_i - \Xi \leq \frac{1}{2} \Delta \quad \forall i = 1, \ldots, N
\end{array} \right)
\]

\[
f(R)
\]

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

\[ f(R) \text{ is not differentiable everywhere...} \]
Solving this problem requires...

- a derivative-free method
- a global search technique

\[ f(R) \text{ has many local minima...} \]
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)
How to solve an optimization problem?

Coming up next...

The problem: minimum-volume OBB

Bringing optimization into the game

How to solve an optimization problem?

- Genetic algorithms (GA)
- The Nelder-Mead algorithm (NM)
- HYBBRID: let’s mix GA and NM together!

Results

Conclusion
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, ...)

C.-T. Chang et al. Fast oriented bounding box optimization on the rotation group $SO(3, \mathbb{R})$
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)

Reflection | Expansion | Contraction | Reduction

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…

Current objective = $-0.23094$

The initial simplex
Nelder-Mead and the six-hump camel back...

Current objective = −0.23094

Iteration 1: contraction
Nelder-Mead and the six-hump camel back...

Current objective = -0.84637

Iteration 2: reflection
Nelder-Mead and the six-hump camel back...

Current objective = -0.84637

Iteration 3: contraction
Nelder-Mead and the six-hump camel back...

Current objective = -0.84637

Iteration 4: contraction
How to solve an optimization problem?

The Nelder-Mead algorithm (NM)

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

Current objective = $-0.84637$

Iteration 5: contraction
Nelder-Mead and the six-hump camel back...
Nelder-Mead and the six-hump camel back...

Current objective = −0.84637

Iteration 7: contraction
Nelder-Mead and the six-hump camel back...

Current objective = −0.86796

Some more iterations...
How to solve an optimization problem?

The Nelder-Mead algorithm (NM)

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

Some more iterations...
Nelder-Mead and the six-hump camel back...

The final result

Current objective = −1.03163
How to solve an optimization problem?

HYBBRID: let’s mix GA and NM together!

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

- **Crossover II:** Create $\frac{M}{2}$ offsprings by affinely combine vertices:
  
  \[ A_1 B_1 C_1 D_1 \otimes A_2 B_2 C_2 D_2 \rightarrow A_3 B_3 C_3 D_3 \text{ with } A_3 = \lambda A_1 + (1 - \lambda) A_2 \]
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:
  \[ A_1 B_1 C_1 D_1 \otimes A_2 B_2 C_2 D_2 \rightarrow A_3 B_3 C_3 D_3 \text{ with } A_3 = \lambda A_1 + (1 - \lambda) A_2 \]
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:
  \[
  A_1 B_1 C_1 D_1 \otimes A_2 B_2 C_2 D_2 \rightarrow A_3 B_3 C_3 D_3 \text{ with } A_3 = \lambda A_1 + (1 - \lambda) A_2
  \]
- **Mutation**: Apply $K$ Nelder-Mead iterations on each offspring
HYBBRID:

Nelder-Mead algorithm ⊕ 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

C.-T. Chang et al.
Coming up next...

The problem: minimum-volume OBB

Bringing optimization into the game

How to solve an optimization problem?

Results

Behaviour of HYBRID
Comparison to other algorithms

Conclusion
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 \( \sim 300 \) 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

![Graph showing comparison between different optimization methods]

**Constrained smooth opti:** success rate \( \sim 40\% \) but mainly AABBs...
Results

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...

The problem: minimum-volume OBB

Bringing optimization into the game

How to solve an optimization problem?

Results

Conclusion
Conclusion

- **HYBBRID**: Nelder-Mead ⊕ 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!
HYBBRID: Nelder-Mead ⊕ 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!