Exact Motion Planning with Rotations: the Case of a Rotating Polyhedron

Mr. Przemysław Dobrowolski

Warsaw University of Technology
Faculty of Mathematics and Information Science

5 July 2012

Problem description

We are given a scene consisting of a number of polyhedral obstacles. There is one selected polyhedron which is allowed to rotate. We will call it a *rotating polyhedron*.

Problem description

We are given a scene consisting of a number of polyhedral obstacles. There is one selected polyhedron which is allowed to rotate. We will call it a *rotating polyhedron*.

A motion planning task

A motion planning task is a problem of finding a continuous collision-free rotation path of a *rotating polyhedron* admist polyhedral obstacles from the selected begin rotation to end rotation.

Problem description

We are given a scene consisting of a number of polyhedral obstacles. There is one selected polyhedron which is allowed to rotate. We will call it a *rotating polyhedron*.

A motion planning task

A motion planning task is a problem of finding a continuous collision-free rotation path of a *rotating polyhedron* admist polyhedral obstacles from the selected begin rotation to end rotation.

Configuration space

It is convenient to solve motion planning problems in terms of a configuration spaces.

Configuration space

It is convenient to solve motion planning problems in terms of a configuration spaces.

Configuration (of a polyhedron)

Any mathematical description which indicates a rotation (of a polyhedron).

We are going to use *spinor* (quaternion) representation and Spin(3) space. Each rotation is described by four numbers.

Configuration space

It is convenient to solve motion planning problems in terms of a configuration spaces.

Configuration (of a polyhedron)

Any mathematical description which indicates a rotation (of a polyhedron).

We are going to use *spinor* (quaternion) representation and Spin(3) space. Each rotation is described by four numbers.

Configuration space

The set of all configurations.

Examples of configuration space:

- $\mathbb{R}^3 x SO(3)$ - polyhedron motion in $\mathbb{R}^3$ with rotations
- $\mathbb{R}^2 x SO(2)$ - polygon motion in $\mathbb{R}^2$ with rotations

Configuration space (II)

Points in a configuration space:

Configuration space (II)

Points in a configuration space:

Empty configuration space

The subset of a configuration space which represent rotations which cause no collision of the rotating polyhedron with obstacles.

Configuration space (II)

Points in a configuration space:

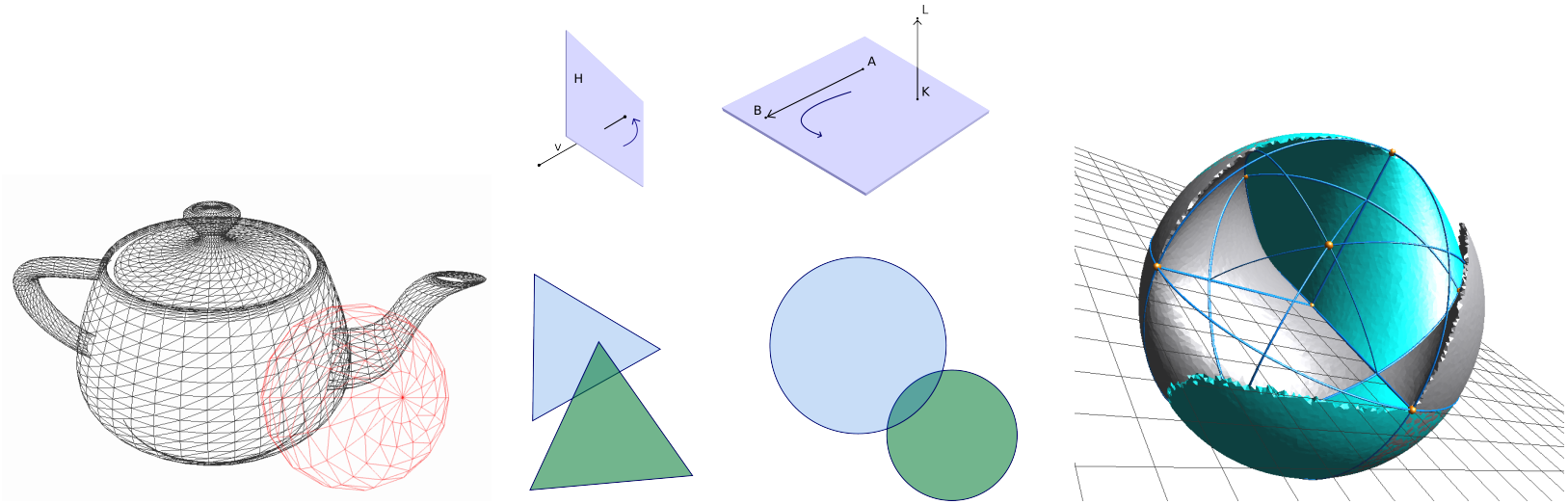
Empty configuration space

The subset of a configuration space which represent rotations which cause no collision of the rotating polyhedron with obstacles.

Forbidden (configuration space

The subset of a configuration space which represent rotations which cause at least one collision of the rotating polyhedron with obstacles.

libcs library: the approach for solving the problem



(a) Scene (b) Predicates (c) C-space

Figure: libcs algorithm overview

libcs library: new theoretical results

Theorem (Exact predicates)

The following predicates can be represented exactly:

- *plane-point, plane-ball collisions*
- *ball-ball collision*
- *triangle-triangle collision*

They are represented by quadrics in Spin(3). Each quadric divides the whole space into a colliding part (forbidden c-space) and non-colliding part (empty c-space).

libcs library: new theoretical results

Theorem (Exact predicates)

The following predicates can be represented exactly:

- *plane-point, plane-ball collisions*
- *ball-ball collision*
- *triangle-triangle collision*

They are represented by quadrics in Spin(3). Each quadric divides the whole space into a colliding part (forbidden c-space) and non-colliding part (empty c-space).

Theorem (Surface intersection in Spin(3))

It is possible to implement surface intersection (and thus, a motion planning algorithm) in SO(3) by using a library for quadric intersection in $\mathbb{P}^3$.

libcs library: new theoretical results

Theorem (Exact predicates)

The following predicates can be represented exactly:

- *plane-point, plane-ball collisions*
- *ball-ball collision*
- *triangle-triangle collision*

They are represented by quadrics in Spin(3). Each quadric divides the whole space into a colliding part (forbidden c-space) and non-colliding part (empty c-space).

Theorem (Surface intersection in Spin(3))

It is possible to implement surface intersection (and thus, a motion planning algorithm) in SO(3) by using a library for quadric intersection in $\mathbb{P}^3$.

libqi - a complete library for quadric intersection

The algorithm

Configuration space construction algorithm:

- collect a list of predicates from scene
- calculate surfaces in configuration space
- intersect all surfaces - edges of the graph
- intersect edges of the graph with all surfaces - vertices of the graph
- collecting adjacency information and final graph construction

Result: intersection graph in configuration space in which we can locate desired motion paths.

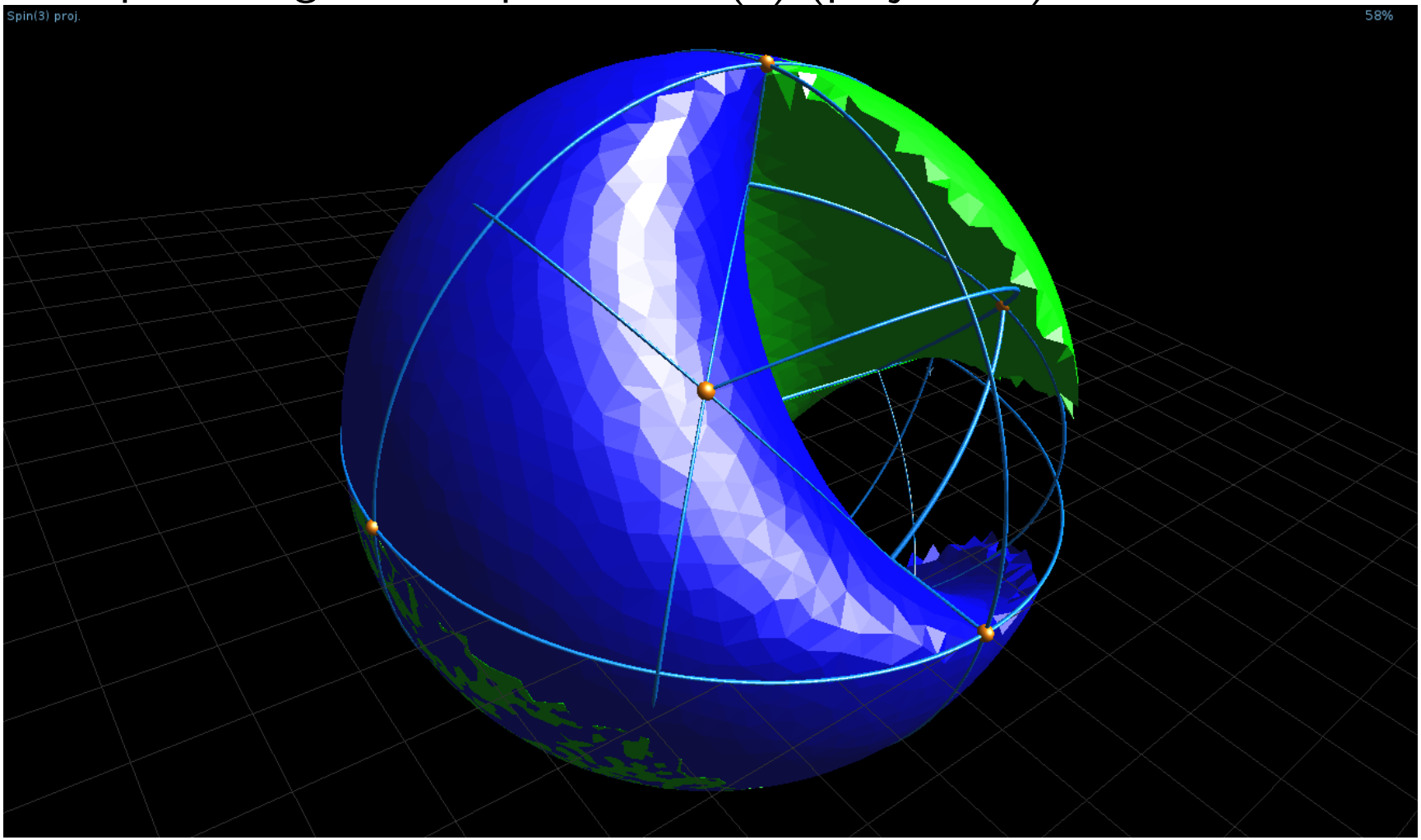
libcs library: implementation

libcs: a library for computing configuration spaces.

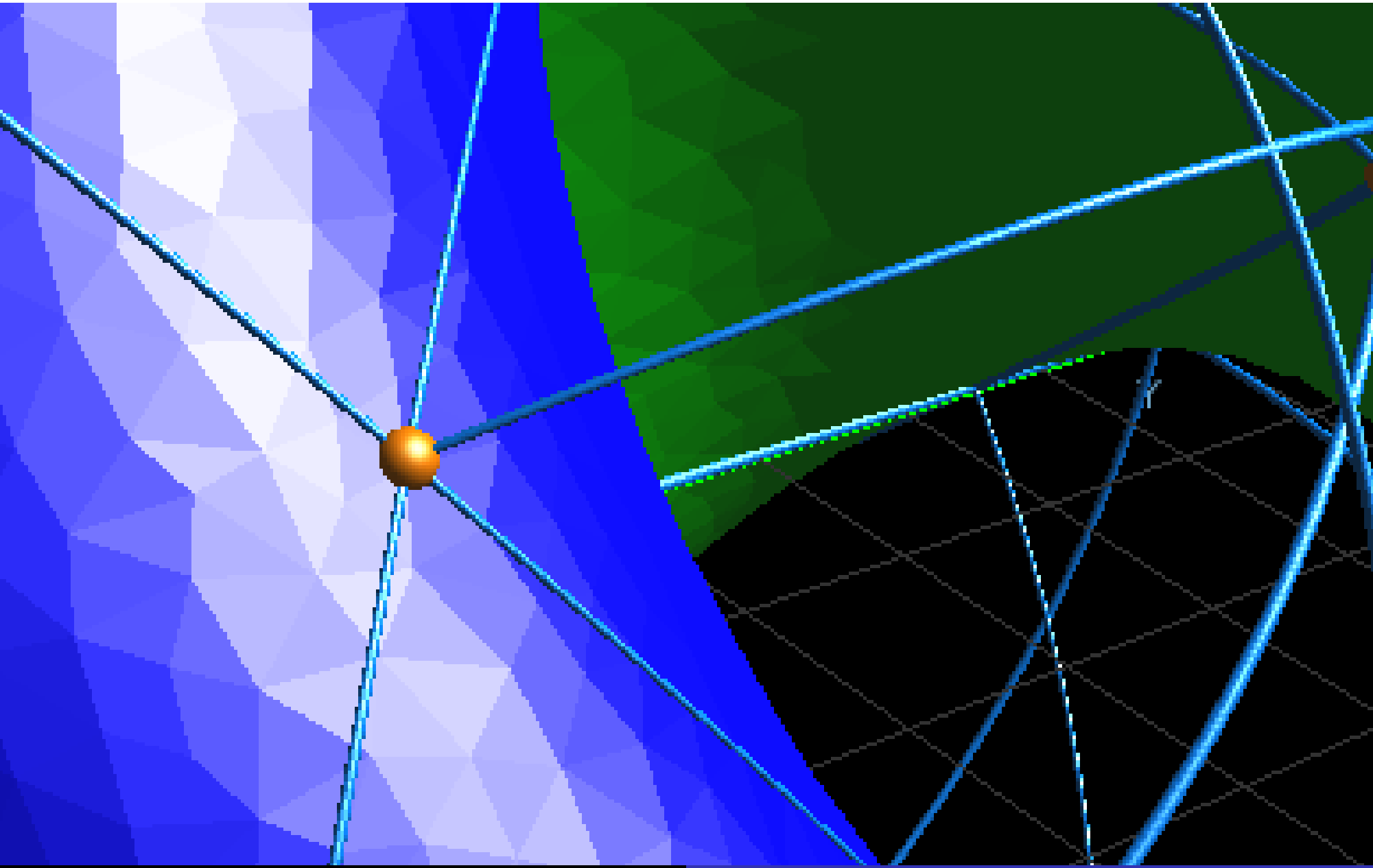
property	value	comments
c-space	Spin(3)	
rot. rep.	spinors	or quaternions
result rep.	graph of intersections	
code	templated C/C++	based on CGAL
arithmetic	gmp Z, double,	provided by CGAL
status	implemented	partially optimized
complexity	$O(m^3 n^3 log(mn))$	output sensitive

The result of algorithm

Example configuration space in SO(3) (projection).



The result of algorithm



Conclusions

- First exact motion planning algorithm involving 3D rotations
- First exact representation of SO(3) configuration space
- An ingredient for other motion planning algorithms (involving 3D rotations)
- algorithms in SO(3) are very different from similar ones in $\mathbb{R}^3$ (difficult localization)
- Near-optimal algorithm and output sensitive

Thank you