The CUIK Suite

Analyzing the motion closed-chain multibody systems

By J. M. Porta, L. Ros, O. Bohigas, M. Manubens, C. Rosales, and L. Jaillet

Many situations in Robotics require the analysis of the motions of complex multibody systems. These are sets of articulated bodies arising in a variety of devices, including parallel manipulators, multifingered hands, or reconfigurable mechanisms, but they appear in other domains too, as mechanical models of molecular compounds or nanostructures. Closed kinematic chains arise frequently in such systems, either due to their morphology, or to geometric or contact constraints to fulfill during operation, giving rise to configuration spaces of an intricate structure. Despite appearing very often in practice, there is a lack of general software tools to analyze and represent such configuration spaces. Existing packages are either oriented to open chain systems, or to specific robot types, which hinders the analysis and development of innovative manipulators. This paper describes the CUIK suite, a software toolbox for the kinematic analysis of general multibody systems. The implemented tools can isolate the valid configurations, determine the motion range of the whole multibody system or of some of its parts, detect singular configurations leading to control or dexterity issues, or find collision- and singularity-free paths between configurations. The toolbox has applications in robot design and programming, and it is the result of several years of research and development within the *Kinematics and Robot Design* group at IRI, Barcelona. It is available under GPLv3 license from `http://www.iri.upc.edu/cuik`.

Motivation and Outlook

The notion of configuration space (C-space) is fundamental in Robotics. It allows designing motion analysis algorithms for broadly-defined classes of robots, without worrying about their particular geometry or multibody structure. In most Robotics courses this notion is introduced for open-chain robots, where the C-space has an explicit global parametrization. In this way, C-spaces are readily understood and algorithms operating on them can be easily defined. In real problems, however, the C-space can have a more complex structure. On a welding robot, for instance, the set of valid configurations reduces to those where the end-effector is in contact with the surface to be welded. This constraint implicitly defines a non-parametric C-space whose analysis is not trivial in general. Similar problems arise in Structural Biology, when analyzing the feasible motions of a molecule, or in Computer-Aided Design, when assembling parts using spatial constraints. The CUIK suite provides effective tools for analyzing the C-spaces arising in such a broad range of applications, a point not properly addressed in the related packages available so far.

In all the considered problems, the valid configurations are implicitly defined by a system of kinematic equations encoding the assembly, task, or contact constraints intervening in the problem, and the goal is to analyze the motion capabilities by finding the solutions of such system. In an extreme case, the valid configurations are isolated points. This is what happens, for example, when solving position analysis problems on parallel or serial manipulators, where one wishes to compute the feasible configurations for given values of the input or output coordinates. Historically, the preferred approach to tackle these problems has been variable elimination: the initial equations are reduced to a resultant univariate polynomial, which is solved using well-established meth-

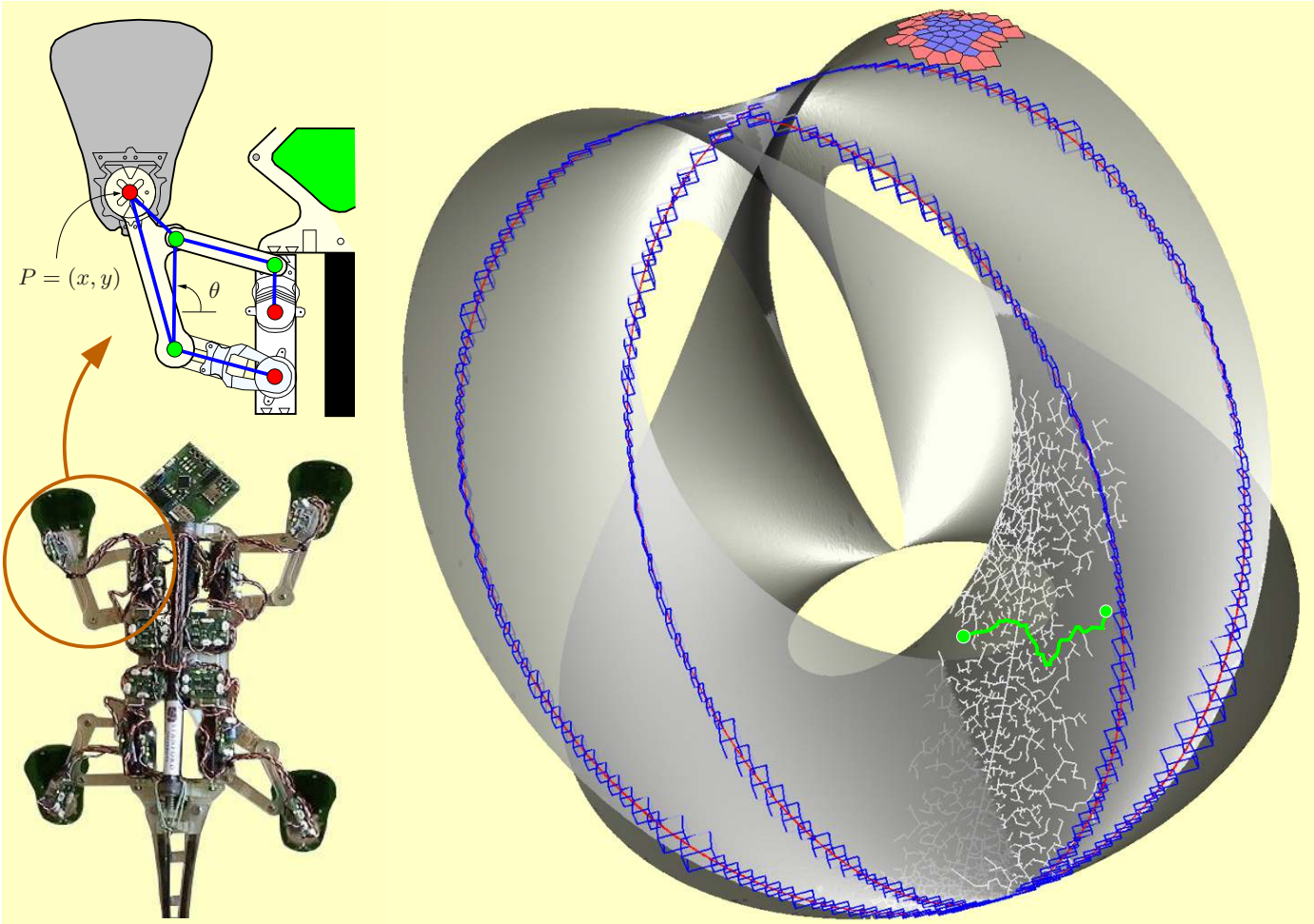

Figure 1. Results of the CUIK suite tools on the Stickybot III robot from the Stanford Biomimetics and Dexterous Manipulation Lab (`http://bdml.stanford.edu`). Each leg has a five-bar mechanism (left, with actuated joints in red), whose C-space is shown projected to the (x, y) coordinates of point P and angle θ (right). The suite allows a comprehensive analysis of the motions of systems of this kind. It can compute the C-space (the translucent gray surface), incrementally produce atlases to solve optimization or path-planning problems (the red and blue mesh at the top), obtain multiresolutive approximations of singularity loci (the blue boxes and the refined red curve), or generate probabilistic roadmaps to connect configurations (the white tree, with a path highlighted in green). The projection of this surface to the (x, y) plane produces the reachable workspace of point P.

ods for this case. The approach is sound, but it may introduce extraneous roots and the size and degree of the resultant rapidly grow with the number of bodies and the complexity of their connection pattern. General elimination packages can be used [8], but they rapidly explode in complexity even on small problems, which explains why no general software for motion analysis has been built using them. The CUIK suite circumvents these issues by adopting an opposite approach. Instead of reducing the initial system of equations to a resultant polynomial, we formulate it as a larger system including only linear and quadratic equations. This particular formulation is then exploited to implement a branch-and-prune method able to compute the whole solution set. The method departs from an initial box bounding this set and effectively prunes unfeasible portions of the box until all solution points are isolated at the desired accuracy. Software toolboxes like ALIAS [16], Bertini [1], or PHC [28] have been applied to kinematics, but they implement general methods for solving systems of algebraic equations. In contrast, the CUIK suite sacrifices generality to gain simplicity and efficiency in the implementation. Opposite to [1, 28], moreover, it directly isolates the real roots instead of the complex ones, which is beneficial in practice.

While many approaches can only identify the solutions when they are isolated points, the branch-and-prune methods in the CUIK suite can also approximate positive-dimensional solutions. This can be used to compute the whole C-space or some of its subsets, providing global information on the motion capabilities of a manipulator. In particular, by adequately defining the equations passed to the solver, the CUIK suite can isolate the workspace boundaries and any of the singularity sets typically defined in the literature [7], becoming the first general tool able to do so, up to the authors' knowledge.

Branch-and-prune methods are complete, in the sense that they obtain *all* configurations, irrespectively of whether they form one or several connected components. In many problems, however, it may be sufficient to explore only those configurations that are path-connected to a given point. To this end, the CUIK suite implements higher-dimensional continuation tools allowing to trace arbitrary manifolds [10]. While branch-and-prune meth-

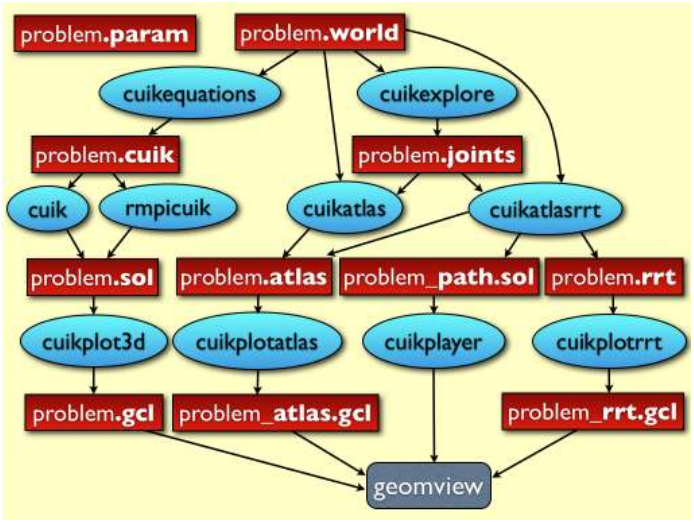

Figure 2. A representative set of the commands implemented in the CUIK suite (ellipses) and their input/output files (rectangles). The parameters file problem.**param** is used by all the commands. The suite implements many other tools not shown here for simplicity. Note that *geomview* is an external visualization program.

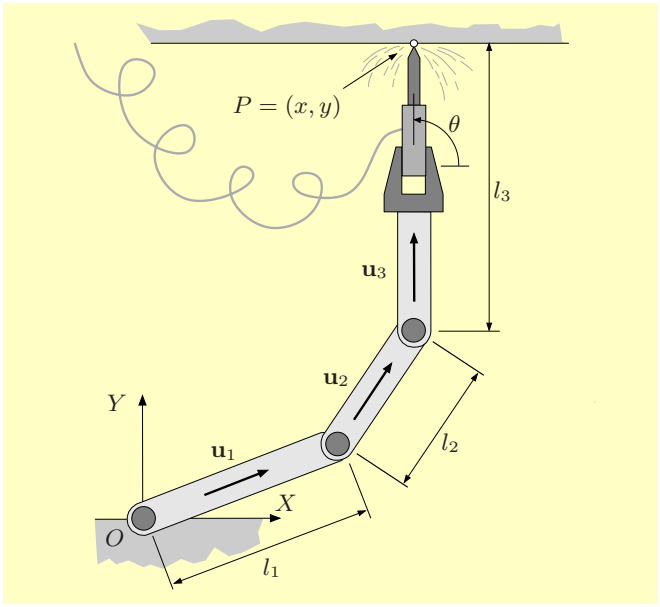

Figure 3. A planar arm welding on a beam. The goal is to weld on a particular point P with orientation θ.

ods proceed by discarding non-valid configurations, continuation techniques march from a given point in all directions identifying new feasible configurations. Local charts of the C-space are constructed and coordinated along the way, defining a global atlas that is suitable to determine feasible motion ranges, optimal configurations, or paths between configurations. Packages to compute the latter exist, but they are oriented to open-chain robots [15, 9, 26, 27], or to specific classes of closed-chain devices [14]. The CUIK suite complements these packages by providing new methods to deal with the general closed-chain case. For C-spaces of moderate dimension, the suite provides strategies to connect start and goal configurations through low-cost, collision- or singularity-free paths. To deal with larger-

```
[constants]
 l1:=1
 l2:=1
 l3:=1.3
 x:=0.25
 y:=2.05
 theta:=pi/2
[system vars]
 u1_x:[-1,1]
 u1_y:[-1,1]
 u2_x:[-1,1]
 u2_y:[-1,1]
 u3_x:[-1,1]
 u3_y:[-1,1]
[system eqs]
 u1_x^2+u1_y^2=1;
 u2_x^2+u2_y^2=1;
 u3_x=cos(theta);
 u3_y=sin(theta);
 l1*u1_x+l2*u2_x+l3*u3_x=x;
 l1*u1_y+l2*u2_y+l3*u3_y=y;
```

Example 1. The *welding0D.cuik* file encoding the problem in Figure 3.

dimensional spaces, the suite implements randomized versions of such tools based on the contruction of rapidly-exploring random trees spanning the C-space.

Fig. 1 shows the main outputs of the CUIK suite on a particular example. In the rest of the paper we describe the methods used to produce such outputs and the associated line commands, illustrating them on a simple tutorial example. A representative set of commands and their input/output files are summarized in Fig. 2. The documentation available on-line describes all the examples and functionalities in thorough detail.

Branch-and-prune methods

A basic example shows how the CUIK suite can address position analysis problems. Consider the planar manipulator in Fig. 3, which has three links of lengths l_1, l_2, and l_3, and three revolute joints. The goal is to compute the arm configurations allowing to weld in the indicated point $P = (x, y)$, with orientation θ. Example 1 shows the equations expressing this inverse kinematics problem in the suite. Observe that the arm configurations are represented using three normalized vectors, `(u1_x,u1_y)`, `(u2_x,u2_y)`, and `(u3_x,u3_y)`, encoding the orientation of each link relative to the X axis of the global frame. The position of the end effector is given by the linear equations in the last two equations, and the `[-1,1]` ranges for the variables define a six-dimensional box bounding the location of all the possible solutions of the problem.

To facilitate the generation of the *cuik* files, the suite allows defining the problems in a high-level form. By executing

```
> cuikequations welding0D.world
```

a set of equations equivalent to that in Fig. 3 is obtained from the *world* file given in Example 2. This file describes the links and joints of the multibody system. A link definition includes the name of the link and its geometric shape, and a joint is given by its type, the two links connected, and the position of the joint

```
[constants]
 l1:=1
 l2:=1
 l3:=1.3
 x:=0.25
 y:=0.75
 theta:=pi/2
[links]
 environment: body "beam.off" blue
 link1:       body "link1.off" gray
 link2:       body "link2.off" gray
 link3:       body "link3.off" gray
              body "gripper.off" black
[joints]
 revolute: ground (0,0,0) (0,0,1)
           link1 (0,0,0) (0,0,1)
 revolute: link1 (l1,0,0) (l1,0,1)
           link2 (0,0,0) (0,0,1)
 revolute: link2 (l2,0,0) (l2,0,1)
           link3 (0,0,0) (0,0,1)
 fix:      environment
           link3
           Tx(x)*Ty(y)*Rz(theta)
```

Example 2. The *welding0D.world* file from which a system of equations equivalent to that in Fig. 3 can be automatically generated.

in the local frame of each link. Fig. 4 shows some of the robots modelled with *world* files in the CUIK suite. These predefined *world* files can be used as a starting point to easily define new problems.

By means of simple manipulations, the equations in Fig. 3 can be simplified into the following system of equations

```
0.5*u2_x + 1.5*u2_y = 0.625,
u2_x^2 + u2_y^2 = 1,
```

and the search box can be limited to the ranges $[-0.75, 1]$ and $[-0.25, 1]$ in `u2_x` and `u2_y`, respectively. The graph of these equations and the new search box are represented using solid and dotted lines in Fig. 5 (top). As shown, the solution points lie in the intersection of a line and a circle in the space of `u2_x` and `u2_y`. To identify such points, the box is first bisected along the `u2_x` axis, obtaining the two yellow boxes in the bottom of the figure. The circular arc inside each yellow box is then approximated by half-planes (shown dashed in the figure). These half-planes are used in conjunction with the linear equation to derive a tiny box around the solution on the left, and a larger dark-gray box bounding the solution on the right. This pruning operation is implemented using linear programming, and it is repeated for each box until no further significant reduction is possible. In the end, if the largest side of the box is below a given threshold, it is considered a solution box. Otherwise it is bisected and the process is recursively applied to the newly-created sub-boxes. The command

```
> cuik welding0D.cuik
```

automates this process. It reads the *cuik* file, simplifies the equations when possible, and applies the pruning and bisection operations until all solution points are isolated at the desired

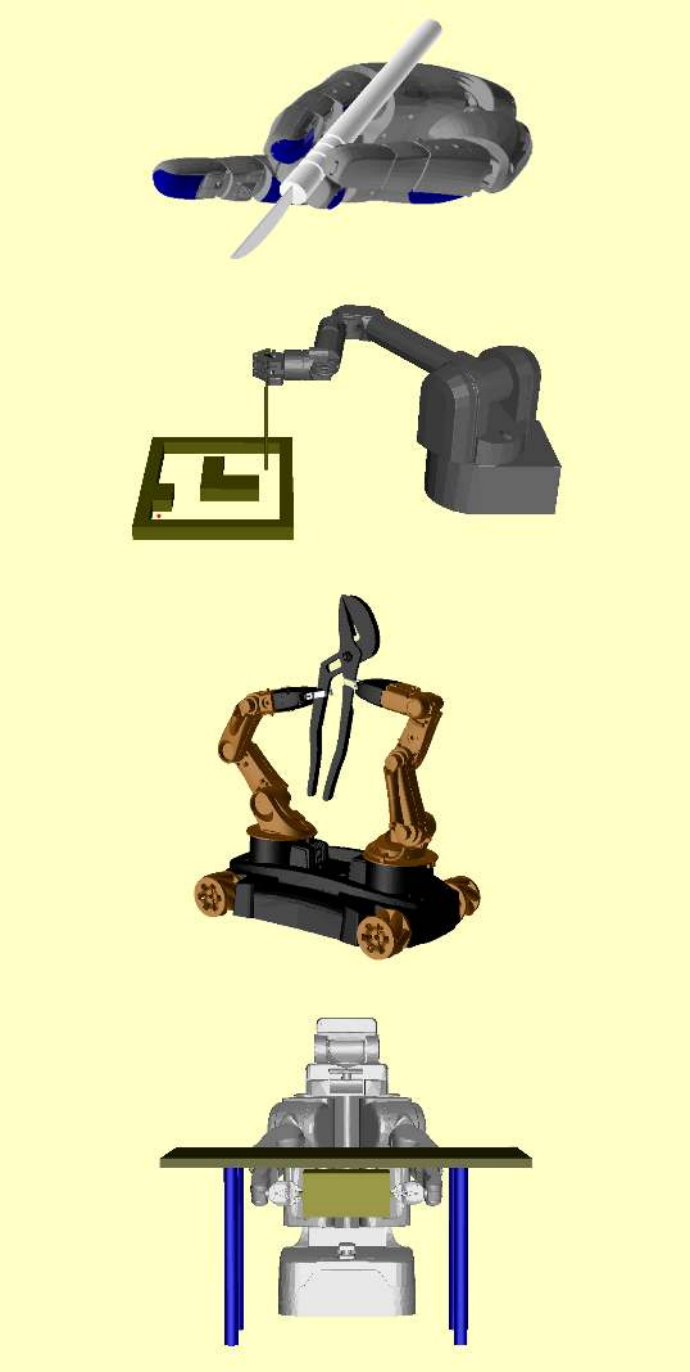

Figure 4. Some of the robots modelled in the CUIK suite: The Schunk anthropomorphic hand, the Barret hand-arm system, the YouBot two-arm mobile platform, and the PR2 service robot.

accuracy. The problem in Fig. 3 is simple, but the method is also successful on complex problems involving general 6R robots and Stewart platforms [21], or in mechanisms with more than 12 chains [20]. To have an idea, elimination or resultant methods are finding their limit in mechanisms of much less complexity [23].

The *cuik* command can also be used to isolate C-spaces of positive dimension. For instance, if the robot in Fig. 3 has to weld along a line in the lower part of the beam while keeping fixed the tool orientation, then the problem exhibits a one-dimensional manifold of solutions. Figure 6 shows several stages of the

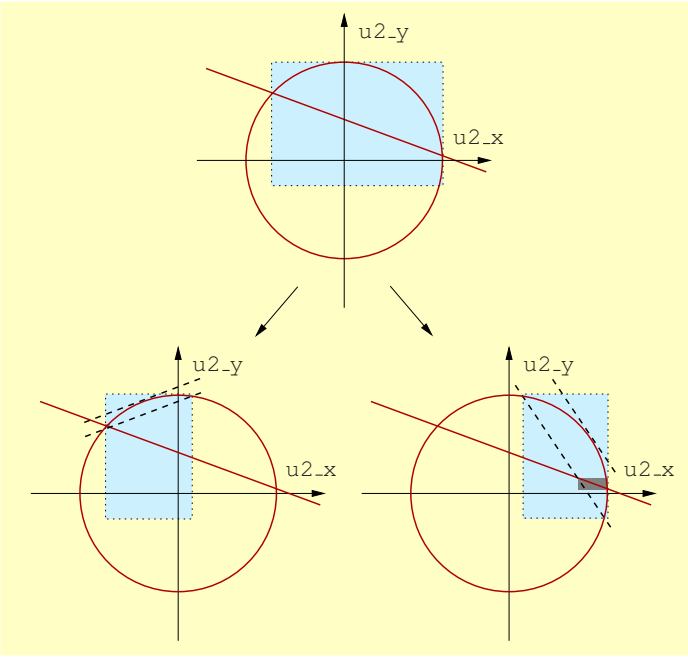

Figure 5. The bisection of the yellow box in the top plot produces the child-boxes in the bottom plots. By considering the linear equation and the relaxed version of the circle equation, we can accurately bound the solution points for the problem encoded in Fig.3.

branch-and-prune method in this case (top plots), and the final box approximation returned as output (bottom plot). The plots are obtained and visualized by executing

```
> cuik welding1D.cuik
> cuikplot3d welding1D.sol 1 3 4 welding1D.gcl
> geomview welding1D.gcl
```

which projects the solution boxes to the subspace of the `u1_x`, `u2_x`, and `u2_y` coordinates. These are the variables appearing in the 1st, 3rd, and 4th positions of the *welding1D.cuik* file describing the problem. The insets in the bottom plot show the arm configurations corresponding to some of the boxes.

If the welding task is further relaxed and the tool can contact the beam with any orientation, the problem exhibits a two-dimensional C-space. Such space can be interactively explored using the command

```
> cuikexplore welding2D.world
```

and it can again be approximated using the `cuik` command, producing the results in Fig. 7 (top). The box approximation includes about 15000 boxes that are computed in just 45 seconds on a standard desktop computer. To solve more difficult problems in reasonable time, the CUIK suite implements a parallel version of the solver, which can be invoked through the `rmpicuik` command. With this tool, the branch-and-prune solver can be executed in large computer grids, where a "master" CPU manages the exploration tree, and a number of "slave" CPUs apply the pruning operations to the assigned boxes. Using this tool on a cluster with 160 CPUs, it took a few minutes to isolate two-dimensional C-spaces like the one in the lower part of Fig. 7. This process would require costly computations using single-CPU machines or alternative approaches [22].

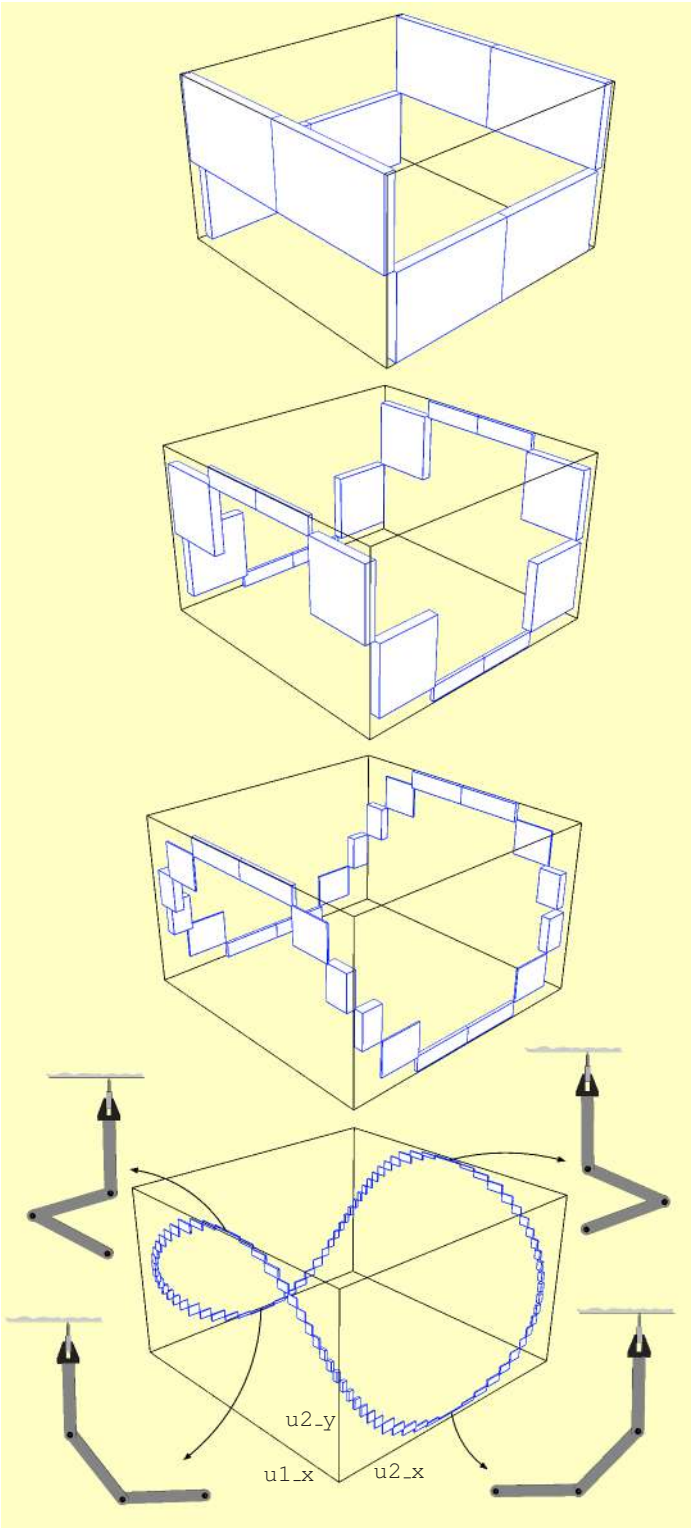

Figure 6. Progress of the branch-and-prune method when computing the C-space of the robot arm in Fig. 3, when the end-effector can contact any point on the lower part of the beam, but with a fixed orientation. Configurations for some of the solution boxes are shown in the bottom picture.

C-spaces of robotic systems typically exhibit singularity subsets. These are loci of configurations where problematic losses of control or dexterity arise [6, 7]. They also reveal the boundaries of the task and joint workspaces, and all motion barriers that may be

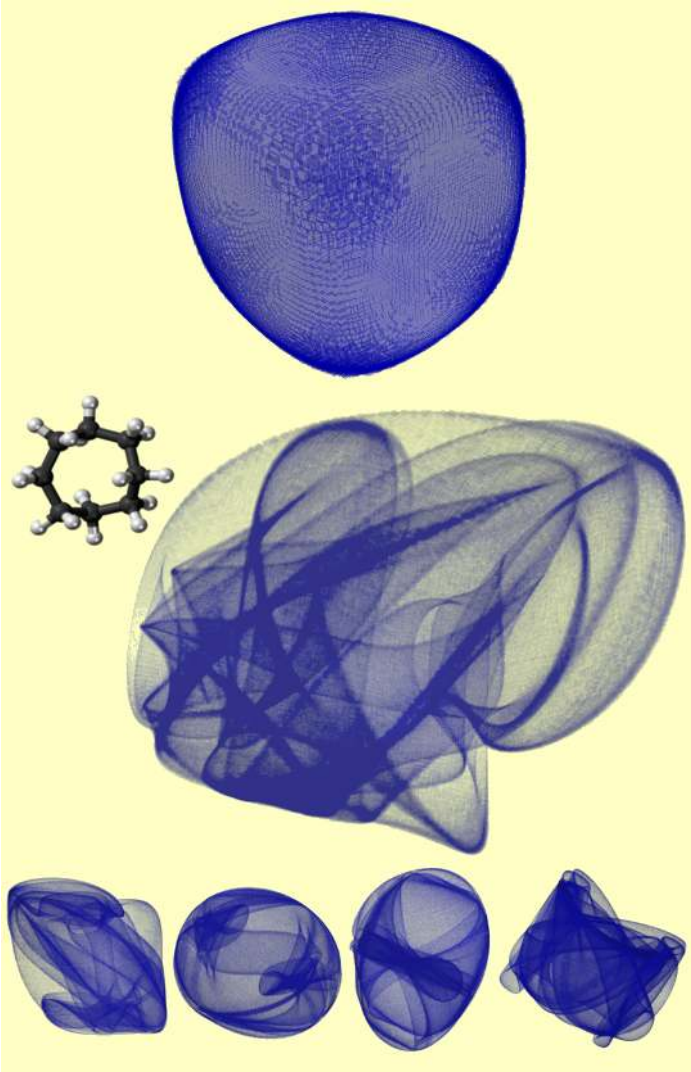

Figure 7. Top: A box approximation of the two-dimensional C-space of the welding task when the tool can contact the lower part of the beam with any orientation. The approximation contains about 15000 boxes, which are here rendered with translucent material. Middle: The cyclooctane molecule, a ring of eight carbon atoms pairwise connected through rotatable covalent bonds and a box approximation of its two-dimensional C-space with about 300000 boxes projected on three of the problem variables. Bottom: Different shapes are obtained by projecting the boxes on alternative triplets of variables.

encountered in their interior [4]. The ability to compute these loci is thus essential, not only to anticipate possible problems during robot operation, but also to provide valuable information to the robot designer. The CUIK suite can be employed to isolate all singular configurations of a manipulator by appropriately formulating their equations and passing them to the solver [2, 7]. For example, Fig. 8 depicts typical forward and inverse singularity loci obtained by the suite tools on a 3-RRR manipulator, shown projected to the pose coordinates of the end-effector, x, y, and θ. The blue surface corresponds to the inverse singularities, which delimit the boundary of the (x, y, θ) workspace. The red surface corresponds to the forward singularities, which indicate the configurations where velocity control issues arise.

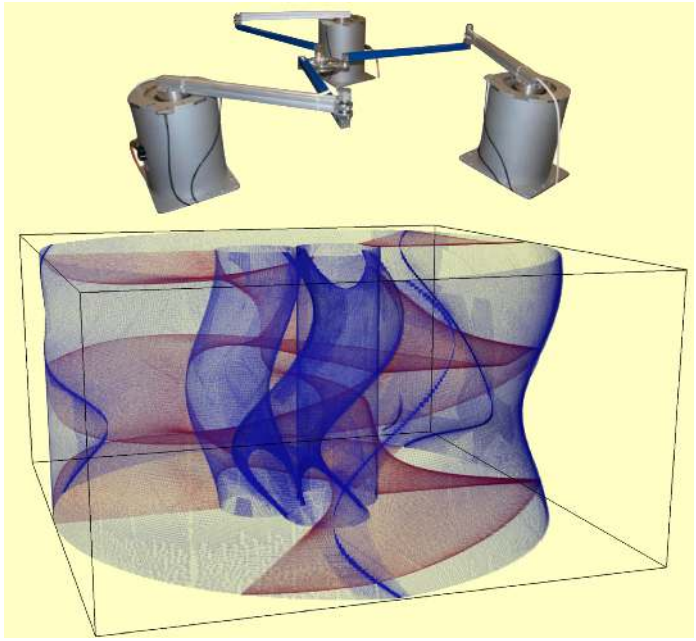

Figure 8. Top: a picture of a 3-RRR manipulator taken from `http://www.imes.uni-hannover.de`. Bottom: typical singularity loci computed for such kind of manipulators.

Continuation methods

Continuation methods generate atlases of the C-space regions that are connected to a given point. To see how such atlases can be constructed, let's assume that $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ is any system of equations encoding the kinematic constraints of the multibody system, whose solution set constitutes the C-space $\mathcal{C}$ under consideration. Also let $\mathbf{x}_i$ be an initial configuration satisfying $\mathbf{F}(\mathbf{x}_i) = \mathbf{0}$. The points in the tangent space of $\mathcal{C}$ at $\mathbf{x}_i$, $\mathcal{T}_i$, can be parametrized by

$$\mathbf{x}'_i = \mathbf{x}_i + \mathbf{\Phi}_i \mathbf{u}\,, \tag{1}$$

where $\mathbf{\Phi}_i$ is a matrix providing an orthonormal basis of $\mathcal{T}_i$, and $\mathbf{u}$ is a parameter vector with the same dimension as $\mathcal{C}$. By choosing a value for $\mathbf{u}$ in Eq. (1) we obtain a point $\mathbf{x}'_i \in \mathcal{T}_i$, which can be projected to $\mathbf{x}_j$, the point in $\mathcal{C}$ lying in the normal line through $\mathbf{x}'_i$, by solving the system

$$\begin{cases} \mathbf{F}(\mathbf{x}_j) = \mathbf{0}, \\ \mathbf{\Phi}_i^\top (\mathbf{x}_j - \mathbf{x}'_i) = \mathbf{0}, \end{cases} \tag{2}$$

as illustrated in Fig. 9-(a). The new configuration $\mathbf{x}_j$ can be used to define a new chart that can be coordinated with the previous chart [Fig. 9-(b)], and the process can be iterated until the whole component of $\mathcal{C}$ reachable from $\mathbf{x}_i$ gets fully covered [Fig. 9-(c)]. Every time a new chart is defined, the CUIK suite checks for the presence of bifurcations of $\mathcal{C}$, and propagates the atlas construction through such bifurcations in order not to leave areas unexplored. The command

```
> cuikatlas welding2D
```

takes the *world* file describing the robot welding problem when the robot can contact the lower part of the beam with any orientation and a *joints* file providing the initial configuration. As output it returns an *atlas* file including the charts, which can

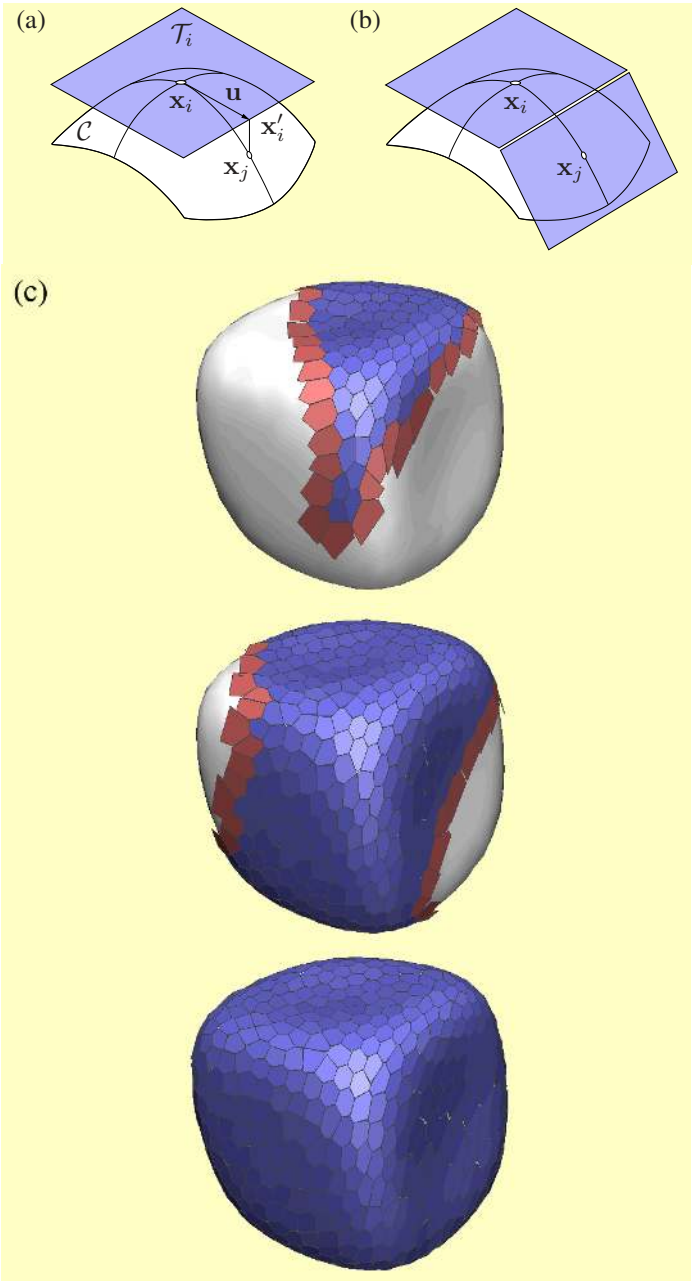

Figure 9. (a) A chart is used to obtain new configurations by projecting points from the tangent space. (b) When a new chart is defined, the chart domains are mutually coordinated to keep track of the explored area. (c) Progress of the atlas construction method on the C-space of the robot arm in Fig. 3, assuming that the welding tool can contact the lower part of the beam with any orientation. This C-space is shown projected on three of the problem variables. Red polygons represent the charts to be extended in subsequent iterations.

be visualized using

```
> cuikplotatlas welding2D 0 9 18
> geomview weld2D_atlas.gcl
```

Here, the indices 0, 9, and 18 indicate the three coordinates on which to represent the atlas. The results of these commands are shown in the lower part of Fig. 9. In this case, the atlas construction takes 0.1 seconds on a standard desktop computer, while the isolation of the same space using branch-and-prune techniques

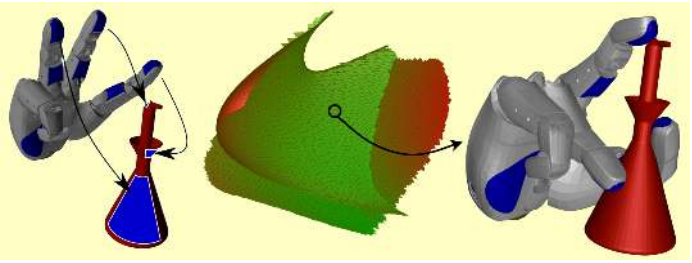

Figure 10. An optimization problem in the context of grasp synthesis. Left: The configuration of the hand-object system must satisfy a number of contact constraints. An initial feasible grasp is computed using the branch-and-prune methods [25] and is used as the initial configuration from which to construct an atlas of the relevant C-space subset [24]. Right: The chart centers of this atlas can be evaluated according to a performance criterion in order to select the optimal grasp. The red and green colors in the atlas correspond to low- and high-quality grasps, respectively.

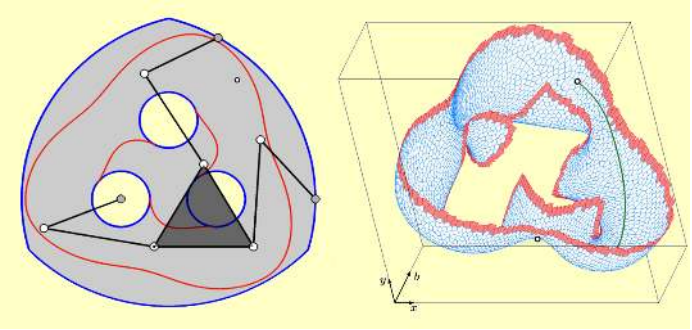

Figure 11. Left: A singularity-free path planning problem on a 3-RRR manipulator [3]. The gray area is the constant-orientation workspace of the manipulator, and the red curves correspond to the forward singularity locus to be avoided during the move. Right: The solution path computed by the CUIK suite, in green, projected on three variables relevant to the problem. The red charts correspond to configurations that are almost singular.

requires about 45 seconds. Such a remarkable speed-up comes at the expense of neglecting other connected components, but allows tackling difficult optimization problems involving large multibody systems (Fig. 10).

To solve path planning problems, the CUIK suite exploits the fact that an atlas implicitly defines a roadmap of the C-space, whose nodes and edges are, respectively, the chart centers and and the collision-free transitions between neighboring charts. The collisions to avoid can be specified in the *world* files. The roadmap can be readily used to resolve multiple planning queries between different configurations. For cases where only one query needs to be resolved, though, it is better to employ the *cuikatlasAstar* tool, which constructs only those charts leading to the shortest path between the given configurations. Based on this tool, singularity-free path planners for general closed-chain [3] and cable-driven manipulators [5] have been developed, solving problems for which no alternative satisfactory solution had been given to date (Fig. 11). Trading off optimality for efficiency, the *cuikatlasGBF* tool provides a path planner that implements a greedy best first strategy to connect the query configurations [19].

Both the *cuikatlasAstar* and the *cuikatlasGBF* tools can be inefficient in cluttered environments, and they do not scale gracefully to higher dimensions. To avoid these weaknesses, the CUIK suite includes the *cuikatlasrrt* tool, which implements

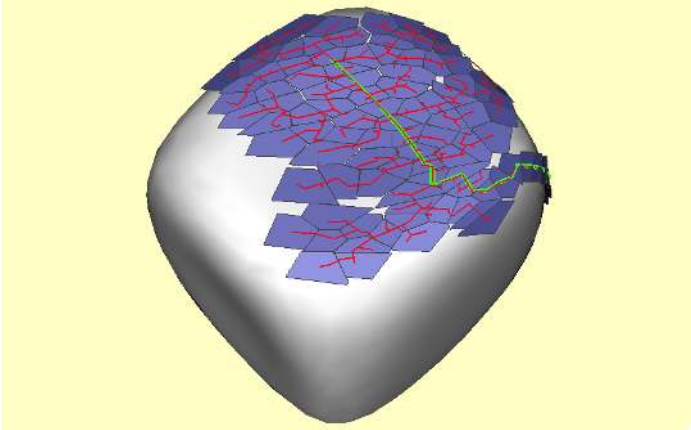

Figure 12. A RRT and a partial atlas solving a path planning query over the C-space manifold of the welding robot when the tool can contact the lower part of the beam with any orientation. In this problem, the robot has to avoid to collide with the beam. The path over the RRT solving the planning query is shown in green.

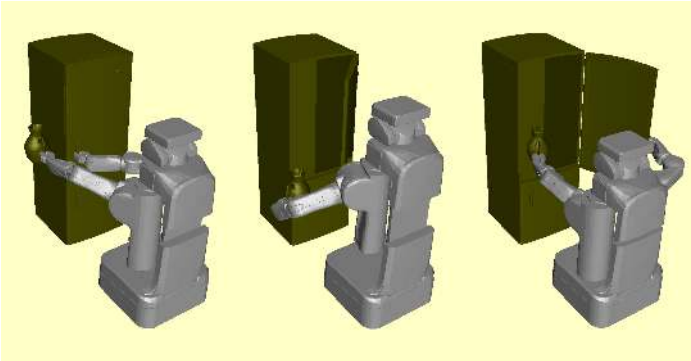

Figure 13. Different stages of the execution of a path allowing to put a pitcher in a fridge that is initially closed.

a sampling method where a partial atlas is used to extend a rapidly-exploring random tree (RRT), which in turn is exploited to decide expansion directions for the atlas [13]. For instance, after specifying the start and goal configurations in the *joints* file, the sequence of commands

```
> cuikatlasrrt welding2D
> cuikplotatlas welding2D 0 9 18
> cuikplotrrt welding2D 0 9 18
> geomview welding2D_rrt.gcl welding2D_atlas.gcl
```

produces an RRT and an atlas like those shown in Fig. 12. The motion of the robot along the path solving the planning problem can be visualized executing

```
> cuikplayer welding2D welding2D_path
```

Using this technique, it is possible to solve problems in pretty high dimensions. Fig. 13, for example, shows three snapshots of the path computed by this method in a manipulation problem involving a PR2 robot. The robot has to carry a pitcher without tilting it, and put it in a fridge that is initially closed. The two arms are open-chain robots, but their configurations are restricted by task and contact constraints, respectively, which make the planning problem rather difficult. The C-space is 8-dimensional, but it takes less than three seconds to determine the solution path.

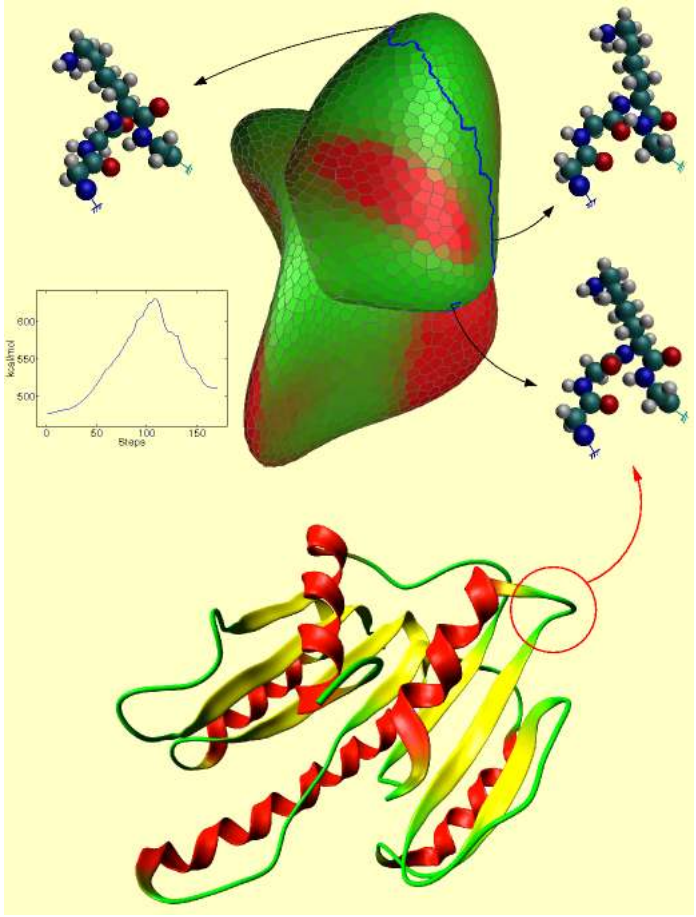

Figure 14. A low-cost path (in blue) computed in the conformational space of a loop of the FTSJ protein of Escherichia Coli (in ribbon diagram in the bottom). The cost is the potential energy of each conformation. The insets show the initial conformation, the transition state (i.e., the conformation with the highest potential energy along the path), and the final conformation. Only the atoms in the loop are shown in such conformations. The plot shows the energy profile along the transition path.

Paths generated with RRT-like algorithms might be costly or unnecessarily long. To address these issues, the CUIK suite includes procedures to generate near-optimal paths when there is a cost function related to the C-space. If the cost is defined for each configuration, one can use the *cuikatlastrrt* tool, which implements an extended version of the T-RRT algorithm in [11]. For instance, Fig. 14 shows a low cost path computed with this method in the case of a short loop of the FTSJ protein of Escherichia Coli, where the cost function is the potential energy of each protein conformation [18]. If the cost is the length of the path, the *cuikatlasrrtstar* tool can be used instead, which is an adaptation of the RRT* asymptotically-optimal path planner to the case of implicitly-defined C-spaces [12].

Conclusions

This paper has described the CUIK suite, a comprehensive set of tools to analyze C-spaces implicitly defined by systems of kinematic constraints. We provided a brief account of the underlying techniques and the commands used to invoke them. Since problems involving kinematic constraints are ubiquitous in Robotics, the suite may potentially be used in contexts beyond

those described in the paper, including mobile robot localization and mapping [17], motion analysis and synthesis of robot formations, tensegrity and deployable structures, or programmable surfaces, to name a few.

In the future we plan to extend the suite to also accommodate the dynamics of multibody systems, in order to facilitate a direct interfacing with robots operating under high accelerations. However, the suite is an open source package under continuous development, and hence we invite the community to use it and to help us to improve it by sending feedback and suggestions on new functionalities to include.

Acknowledgments

The branch-and-prune methods in the CUIK suite have evolved from years of collaboration with Federico Thomas, whose guidance and inspiring comments have certainly shaped the final result. We would like to express our gratitude to Michael. E. Henderson, for introducing the authors to the higher-dimensional continuation methods, and to Dimiter Zlatanov, for his feedback and support on the analysis and interpretation of mechanism singularities.

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness under project DPI2010-18449, by a Juan de la Cierva fellowship supporting Montserrat Manubens, and by a CSIC JAE-Doc fellowship partially funded by the ESF supporting Léonard Jaillet.

References

[1] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler. The Bertini software. `http://www3.nd.edu/~sommese/bertini`.

[2] O. Bohigas. *Numerical Computation and Avoidance of Manipulator Singularities*. PhD thesis, Universitat Politècnica de Catalunya, 2013. Available through the web address `http://goo.gl/wlR0i`.

[3] O. Bohigas, M. E. Henderson, L. Ros, M. Manubens, and J. M. Porta. Planning singularity-free paths on closed-chain manipulators. *IEEE Transactions on Robotics*, 29(4):888–898, 2013.

[4] O. Bohigas, M. Manubens, and L. Ros. A complete method for workspace boundary determination on general structure manipulators. *IEEE Transactions on Robotics*, 28(5):993–1006, 2012.

[5] O. Bohigas, M. Manubens, and L. Ros. Navigating the wrench-feasible C-space of cable-driven hexapods. In *International Conference on Cable-Driven Parallel Robots*, pages 53–68, 2012.

[6] O. Bohigas, M. Manubens, and L. Ros. Singularities of non-redundant manipulators: A short account and a method for their computation in the planar case. *Mechanism and Machine Theory*, 68:1–17, 2013.

[7] O. Bohigas, D. Zlatanov, L. Ros, M. Manubens, and J. M. Porta. A general method for the numerical computation of manipulator singularity sets. *IEEE Transactions on Robotics*, 2013. In press.

[8] D. Cox, J. Little, and D. O'Shea. *An introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 3rd edition, 2007.

[9] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning library. *IEEE Robotics Automation Magazine*, 19(4):72–82, 2012.

[10] M. E. Henderson. Multiple parameter continuation: Computing implicitly defined k-manifolds. *International Journal of Bifurcation and Chaos*, 12(3):451–476, 2002.

[11] L. Jaillet, J. Cortés, and T. Siméon. Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics*, 26(4):635–646, 2010.

[12] L. Jaillet and J. M. Porta. Efficient asymptotically-optimal path planning on manifolds. *Robotics and Autonomous Systems*, 61(8):797–807, 2013.

[13] L. Jaillet and J. M. Porta. Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE Transactions on Robotics*, 29(1):105–117, 2013.

[14] LAAS-CNRS. Move 3D. `http://www.openrobots.org/wiki/move3d`.

[15] S. LaValle. The Motion Strategy Library. `http://msl.cs.uiuc.edu/msl`.

[16] J.-P. Merlet. The Alias software. `ftp://ftp-sop.inria.fr/coprin/ALIAS`.

[17] J. M. Porta. CuikSLAM: A Kinematic-based approach to SLAM. In *IEEE International Conference on Robotics and Automation*, pages 2425–2431, 2005.

[18] J. M. Porta and L. Jaillet. Exploring the energy landscapes of flexible molecular loops using higher-dimensional continuation. *Journal of Computational Chemistry*, 34(3):234–244, 2013.

[19] J. M. Porta, L. Jaillet, and O. Bohigas. Randomized path planning on manifolds based on higher-dimensional continuation. *The International Journal of Robotics Research*, 31(2):201–215, 2012.

[20] J. M. Porta, L. Ros, T. Creemers, and F. Thomas. Box approximations of planar linkage configuration spaces. *ASME Journal of Mechanical Design*, 127:397–405, 2007.

[21] J. M. Porta, L. Ros, and F. Thomas. A linear relaxation technique for the position analysis of multiloop linkages. *IEEE Transactions on Robotics*, 25:225–239, 2009.

[22] J. M. Porta, L. Ros, F. Thomas, F. Corcho, J. Cantó, and J.-J. Pérez. Complete maps of molecular-loop conformational spaces. *Journal of Computational Chemistry*, 29(1):144–155, 2008.

[23] N. Rojas and F. Thomas. Closed-form solution to the position analysis of Watt–Baranov trusses using the bilateration method. *ASME Journal of Mechanisms and Robotics*, 3(3):031001, 2011.

[24] C. Rosales, J. M. Porta, and L. Ros. Grasp optimization under specific contact constraints. *IEEE Transactions on Robotics*, 29(3):746–757, 2013.

[25] C. Rosales, L. Ros, J. M. Porta, and R. Suárez. Synthesizing grasp configurations with specified contact regions. *The International Journal of Robotics Research*, 30(4):431–443, 2011.

[26] M. Saha. The Motion Planning Kit. `http://ai.stanford.edu/~mitul/mpk`.

[27] N. Vahrenkamp. Simox. `http://simox.sourceforge.net`.

[28] J. Verschelde. The PHC software. `http://homepages.math.uic.edu/~jan/PHCpack/phcpack.html`.

Josep M. Porta*, *Lluís Ros*, *Oriol Bohigas*, *Montserrat Manubens*, *Carlos Rosales*, and *Léonard Jaillet, Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens i Artigas, 4-6, 08028, Barcelona, Spain. E-mails: {porta, lros, obohigas, mmanuben, crosales, ljaillet}@iri.upc.edu.