Generic robotic kinematic generator for virtual environment interfaces

Lorenzo Flückiger[*], Laurent Piguet[**], Charles Baur[*]

[*] Swiss Federal Institute of Technology
EPFL, IMT / DMT - 1015 Lausanne, Switzerland

[**] Fourth Planet, Inc. - Santa Clara, CA 95054
Formerly:
NASA Ames Research Center - Intelligent Mechanisms Group
Moffet Field, CA 94301

flueckiger@dmt.epfl.ch

ABSTRACT

The expansion of robotic systems' performance, as well as the need for such machines to work in complex environments (hazardous, small, distant, etc.), involves the need for user interfaces which permit efficient teleoperation. Virtual Reality based interfaces provide the user with a new method for robot task planning and control: he or she can define tasks in a very intuitive way by interacting with a 3D computer generated representation of the world, which is continuously updated thanks to multiple sensors fusion and analysis.

The Swiss Federal Institute of Technology has successfully tested different kinds of teleoperations. In the early 90's, a transatlantic teleoperation of a conventional robot manipulator with a vision feedback system to update the virtual world was achieved. This approach was then extended to perform teleoperation of several mobile robots (Khepera, Koala) as well as to control microrobots used for microsystems' assembly in the μm range.

One of the problems encountered with such an approach is the necessity to program a specific kinematic algorithm for each kind of manipulator. To provide a more general solution, we started a project aiming at the design of a "kinematic generator" (CINEGEN) for the simulation of generic serial and parallel mechanical chains.

With CINEGEN, each manipulator is defined with an ascii file description and its attached graphics files; inserting a new manipulator simply requires a new description file, and none of the existing tools require modification. To have a real time behavior, we have chosen a numerical method based on the pseudo-jacobian method to generate the inverse kinematics of the robot. The results obtained with an object-oriented implementation on a graphic workstation are presented in this paper.

Keywords: robot manipulators, general kinematic chains, inverse kinematics, user interface, virtual environment.

1. INTRODUCTION

Humans design and build robots to assist or substitute them in a large range of tasks. This leads to machines with incredible power and features, which can perform more and more sophisticated tasks. But too many times, the difficulty for the user to command the robot, increases with the complexity of the system. Virtual Reality (VR) based interfaces provide a powerful tool to overcome this problem.

This paper focuses on a "general kinematic generator" which aims at simulating robots in Virtual Environment. Such a tool has appeared as a real need for several research labs as well as industrial companies involved in robotics. Two research labs, both working in the robotic field and using Virtual Reality tools for efficient teleoperation have joined their efforts to achieve such a kinematic generator.

1.1. Background

The Micro-Engineering Department (Institut de Microtechnique: IMT) of the Swiss Federal Institute of Technology (EPFL) is involved in robotic design and development, with a special focus on industrial applications. The experience gained with industrial partners showed that classical methods for robotic systems programming (off-line as well as on-line) usually lack on user-friendliness and performance. This is why since 1990 the IMT is developing VR interfaces to simplify robot task definition.

Rather than writing down complicated code, the user can interact with a 3D model of a robot in a virtual world. The user is able to define high level tasks (like move to this point or grab this object) in just one hand movement or a button press. The virtual world is continuously updated, thanks to different sensors (camera, scanner, etc.), with all detected objects around the robot. After a simulation stage to check all the potential problems, a concrete task can be achieved by sending the automatically generated code to the robot.

Applying this approach, IMT has successfully tested different kinds of teleoperation[1, 2, 3]:
- Transatlantic teleoperation (with standard network) of a 5-axis Mitsubishi robot with a two cameras vision system.
- Path planning for different mobile robots (Khepera & Koala), with virtual world update by proximity sensors.
- Micro-systems control in the μm range including vision feedback with a camera mounted on an optical microscope.

The Intelligent Mechanisms Group (IMG) of the NASA Ames Research Center has developed over the past five years an architecture to operate science exploration robots. Remotely operating complex robotic systems becomes nearly impossible when either the number of degrees of freedom (d.o.f.) to control is too large, or the communication time delay excesses several seconds. In oder to improve current methods for remote operations, the IMG has developed a new user interface, "Virtual Environment Vehicle Interface" (VEVI) to control complex mechanisms and visualize their associated data in a very intuitive way. VEVI heavily relies on Virtual Environment, and provides through real-time 3D graphics display/sensors a powerful tool for mission planning and surveying. It allows to manage all kinds of mobile robots moving in unstructured natural environment, and the interface can be shared by multiple users in distant control sites.

The IMG has proved the usefulness and efficiency of VEVI during several field missions incorporating different kinds of vehicles[4, 5]:
- Underwater vehicles with the Telepresence Remotely Operated Vehicle (TROV) mission under the sea ice near McMurdo Science Station, Antarctica.
- Eight-legged robot with Dante II during a crater descend in Mt. Spurr, Alaska.
- Wheeled vehicle such as Marsokhod during its recent science mission in February 1995 at the Kilauea Volcano in Hawaii.

1.2. Requirements for a new tool

Both the IMG and the IMT have a broad experience with the teleoperation of robots, and were using similar tools for their interfaces. Thanks to a researcher exchange, a collaboration started to fill in a common lack for robot arm control.

The need of a more general tool to build interfaces to control any kind of new robot arms was obvious. The CINEGEN project began to provide a solution to this problem.

The first part of this paper explains the goals and requirements of the "general kinematic simulator". The second part focuses on the implementation of CINEGEN. Finally, some results already obtained are presented.

2. DESIGN

Using Virtual Environments (VE) for the control of complex mechanisms is a great improvement in the user interface domain, since it is the fusion of two tools:
- Mission surveying: VE is a great tool for monitoring a teleoperation. For example, VEVI is able to visualize both concrete data (like current position and state of the robot) and abstract data (like a temperature through different colors).
- Task definition: rather than pilot a robot through a flat classic "control panel", the user is able to define movement and high level tasks in a virtual 3D world, which shows him the behavior of the system in an intuitive way.

While data representation in a VE needs appropriate encoding and transforms to display them in a useful way, controlling a robot arm requires a non-trivial process: calculating the inverse kinematics of the robot in real time.

Actually, a more efficient way for an operator to pilot a robot is to move the adequate part of the robot (usually the end-effector) in a cartesian coordinate system easy to comprehend (often relative to the user point of view). This user input must be converted in terms of robot's actuators input in order for the robot to achieve the desired movement. This implies two types of calculation: first the trivial one, which transforms the user input (relative for example to the point of view) into a coordinate frame attached to the robot. The second, much more difficult to achieve, transforms coordinates expressed in a cartesian space into angular coordinates relative to each robot part (see Figure 1). That's the main purpose of CINEGEN.

2.1 User Interface

CINEGEN is first thought as a powerful tool to control manipulators. It must allow any non-specialist (neither in kinematic algorithms, nor in robot control) to successfully manoeuver different mechanisms in a VE. Providing the user such a tool implies a real time behavior of the robot, as well as adequate input devices.

Interacting with the robot means that the operator can pick any part of the robot and move it (in the general sense: translations and rotations) wherever he wants, as easily as a "drag and drop" in a drawing program. At this level, a robot arm can be seen as a set of geometrical objects with constrained properties. When the user modifies the position of one of them, all the others have to resolve the appropriate position according to all the constraints. This is the work of the kinematic algorithm (see Figure 2), and this task must be done in real time in order to give the user an acceptable visual feedback. In addition to an easy control of a robot, we want to provide the user an easy way to create a virtual robot. This can be achieved if the inverse kine-

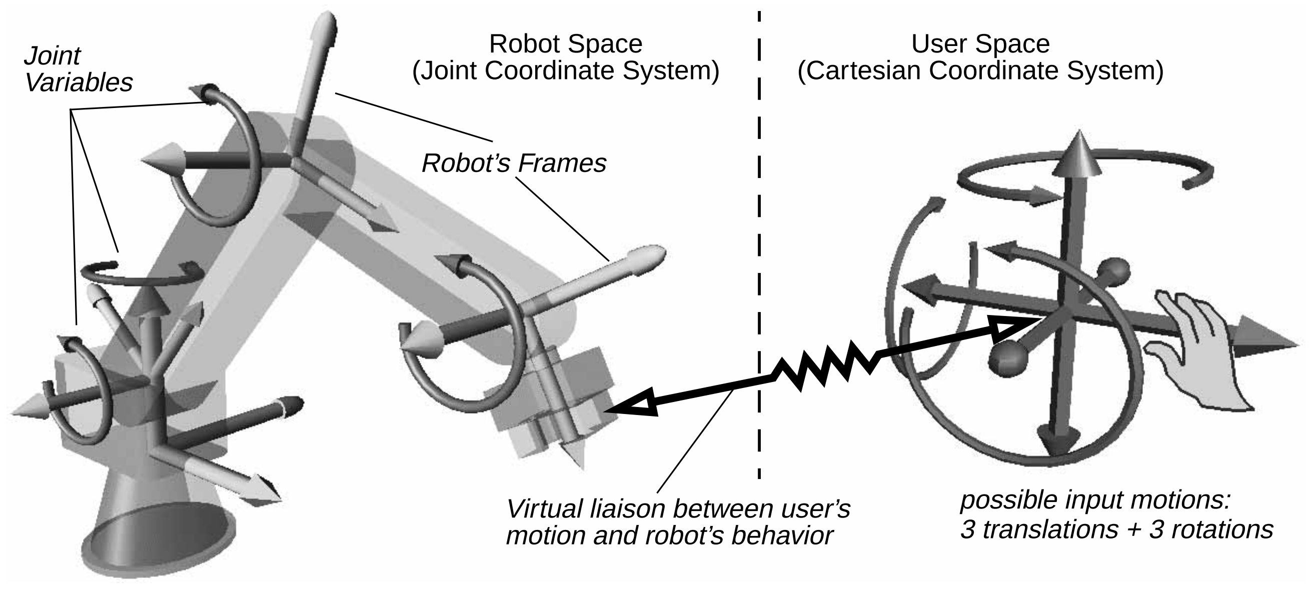

Joint Variables
Robot Space (Joint Coordinate System)
User Space (Cartesian Coordinate System)
Robot's Frames
Virtual liaison between user's motion and robot's behavior
possible input motions: 3 translations + 3 rotations

Figure 1: Relationship between different coordinates systems

matics of a new robot is automatically generated by the program. The user only has to describe the geometric parameters of the manipulator (such as distances between joints). These parameters are stored in a simple text file which describes the robot as well as its potential interactions with the user. At execution, the program reads this file and dynamically builds the robot and its associated inverse kinematics. No recompilation of the program, or linking with new libraries is required.

This description file, named "Robot Arm File Format" (RAFF)[4], is a crucial part of the general kinematic simulator, since it is the link between the user knowledge and the computer algorithms used to generate the virtual environment. This way, a RAFF file format must be designed to be easily understood from both the user and the visualization program, as well as to be automatically generated from external software. For example, we would be able to generate in a simple way this file from a 3D-CAD program, which is used to design the parts of the robot.

2.2 General kinematics

The IMT uses and designs many different robots arms, including serial as well as parallel manipulators. In addition, robot arms mounted on the IMG's vehicles can be changed in function of the mission.

Therefore, CINEGEN aims at controlling any kind of robot manipulator. With such a tool we can easily pilot existing arms as well as study new kinematic structures and their interaction with a specific environment.

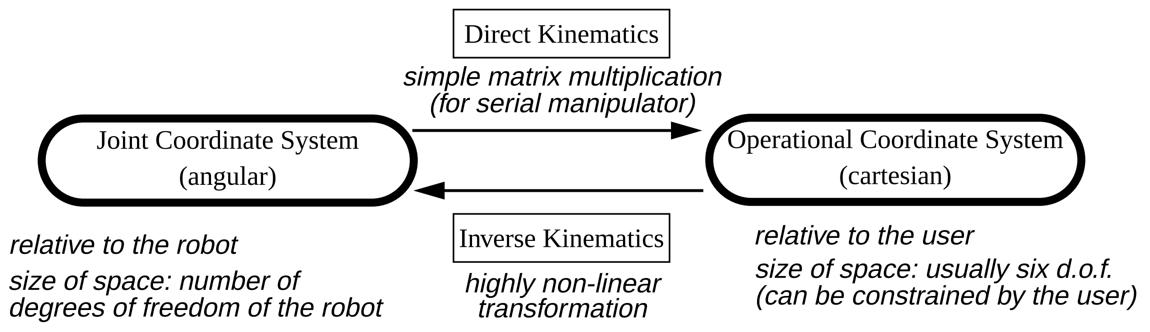

Direct Kinematics
simple matrix multiplication (for serial manipulator)
Joint Coordinate System (angular)
Operational Coordinate System (cartesian)
Inverse Kinematics
highly non-linear transformation
relative to the robot
size of space: number of degrees of freedom of the robot
relative to the user
size of space: usually six d.o.f. (can be constrained by the user)

Figure 2: Caracteristics of the transformations for robot control.

The inverse kinematics of robots is an issue studied for a long time but still remains very complex. Actually, the kinematic equations governing a robot (even a simple serial arm) come from either loop closure equations or joint constraint equations. This mathematical description implies trigonometrical functions and their products, leading to highly non-linear equations.

Several mechanical structures are well-known (kinematic, workspace, singularities), but the general case remains a challenge. Different classes of methods[6, 7] to solve a non-linear set of robot equations exist. In addition to the "hand-made" equation (which are not implementable in a generic manner) for inverse kinematics of a robot, algebraic solutions can be found using elimination methods[8]. Iterative procedure like the widely used continuation method[9, 10] can lead to all numerical solutions for a kinematic problem. However, all these methods are inefficient when the number of equations becomes larger or can have strange numerical behavior, not correlated with the mechanical parameters. It also exists a number of other methods like the Gröbner basis[11] or neural network approaches[12, 13], but they are limited to kinematic problems with a few degrees of freedom.

In the CINEGEN project, the fundamental requirement is real-time inverse kinematics of any kind of robot manipulator. However, the context of the VE interface can simplify the problem regarding the following facts:
- The robot configuration is known at the beginning of the process, and then at any time during the simulation. There is a continuity in robot position, which eliminates the need to find all acceptable configurations.
- High precision positioning is not needed since the goal of CINEGEN is to show a robot behavior at the human level but is not to replace a robot controller, although it can help to optimize the control.

This allows the use of a complete numerical method based on a linearization of the equations and working by incremental changes. The coefficient of the equations will be automatically generated by an algorithm which can deal with any kind of robot manipulator.

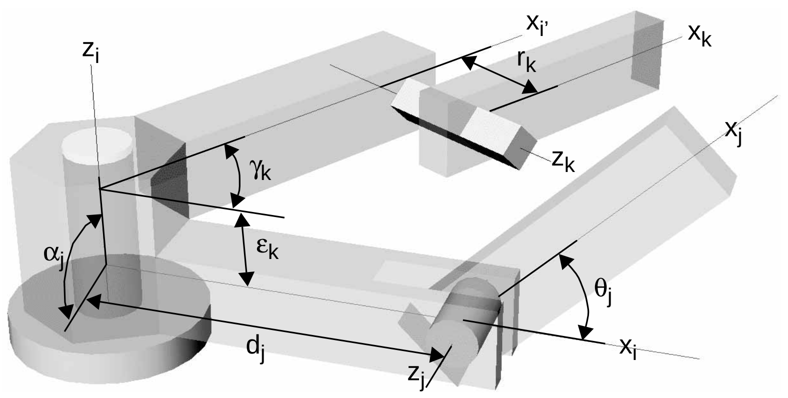
3. IMPLEMENTATION

As described before, the first requirement of CINEGEN for the implementation is real time behavior. In addition, following the VEVI spirit, CINEGEN must be highly modular in order to be usable in different kind of situations. Finally, through its research aspect, CINEGEN tries to keep the maximum expendability, in order to allow further developments. In view of theses facts, the C++ language was chosen for the core of the program. The object-oriented paradigm is well adapted in our application since we can describe each part of the robot as one object with its own properties and behavior. Even if C++ is not a perfect OO language, a lot of very useful libraries are available, and maintains very good efficiency for intensive computations.

3.1 Robot description

3.1.1 Notations

To solve the constraints generated by the kinematic chain of the robot, we must have a mathematical description of the spatial transformations between each joint. Several symbolic notations for mechanisms have already been proposed. The classic Denavit-Hartenberg (D-H) [14] notation, or its Paul [15] version, are still widely used due to their usefulness and clarity. But they both lead to ambiguities for kinematic chains with more than one single branch. The Sheth-Uicker (S-U) [16] method extends the D-H notation for multiple loop kinematic chains in the general case, but is much more complicated to use due to the number of coordinate systems added (two per link for a simple serial chain).



Construction rules:
Z_i = axis of joint i
X_i = axis of common perp. at Z_i & Z_{i+1}

Fixed parameters for link i: d_i & α_i

Joint variables for link i:

prismatic joint:	r_i
rotational joint:	θ_i

Parameters for joint fork: ε & γ

Figure 3: Kleinfinger notation for links description

We chose a third method illustrated in Figure 3 for the links description, introduced by Kleinfinger[17], which presents the following advantages:
- Usable for all serial, treelike or closed loop kinematic chains.
- As simple as the D-H notation in the serial case.
- Less parameters than the S-U notation in complex cases.

This notation permits with a maximum of 6 parameters for each link (3 distances + 3 angles) to describe any kind of kinematic chain.

3.1.2 Structure

In order to simulate any kind of robot arm, keeping simplicity, the kinematic chain is always described as a tree of links. For serial arms, the tree contains only one branch, while for multiple end-effector arms, it has several ones. Closed loops mechanisms are also represented as trees: each loop is “broken” at a user-defined joint and new constraints are added to re-close this loop during the simulation (see Figure 4).

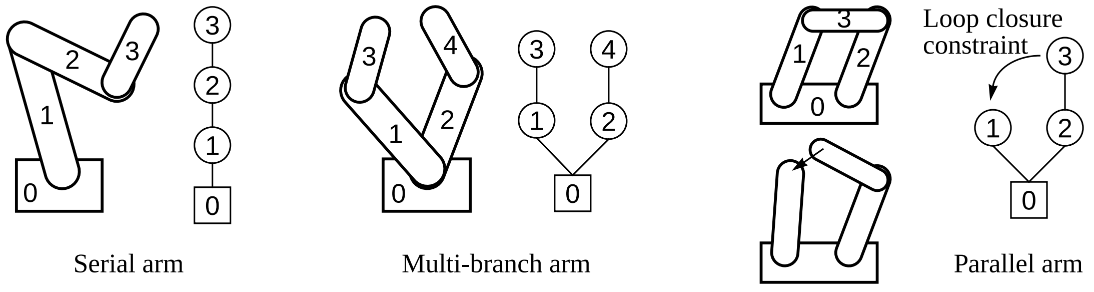
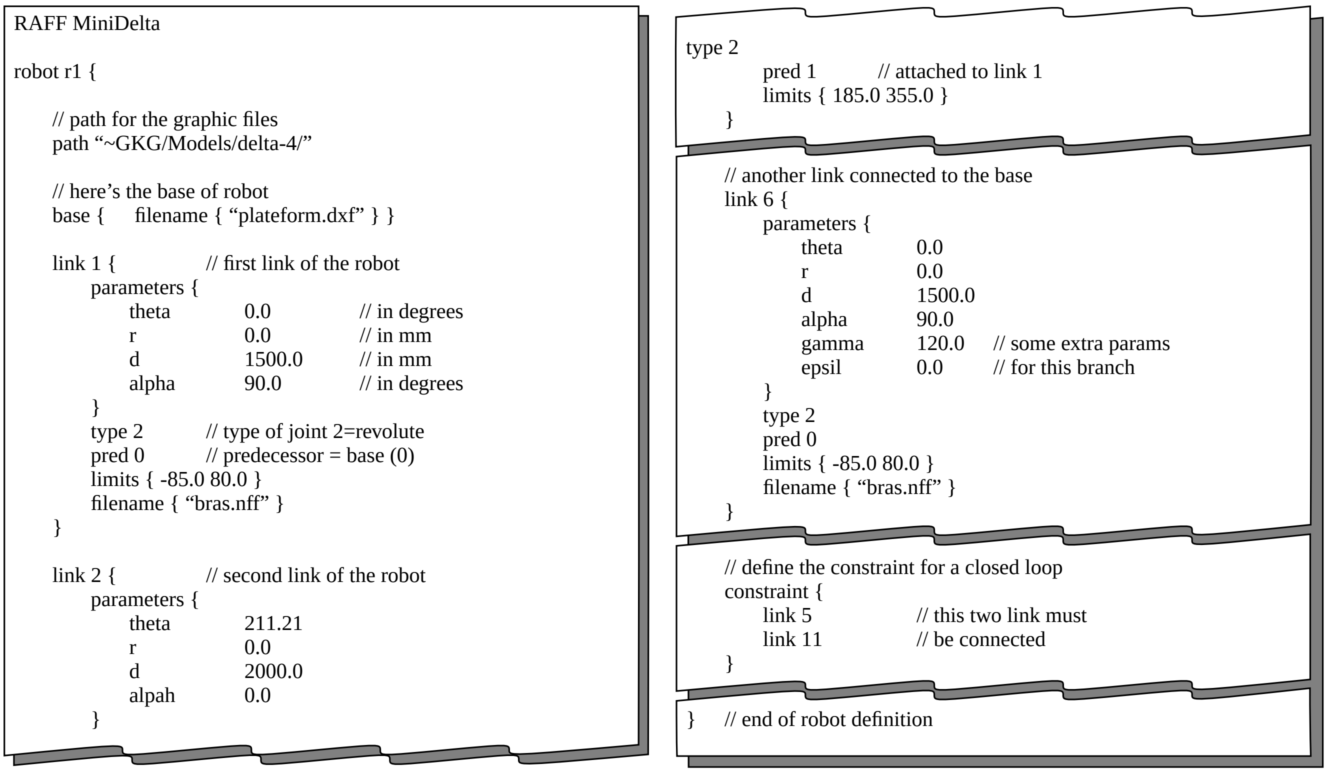


Figure 4: Treelike representation of different kind of robot arms

This approach permits to have a single general algorithm for the inverse kinematics calculation, regardless of the kind of robot arm. In addition, describing the robot structure is very easy: only one number is necessary to identify the father of each link, and some additional constraints are described for the closed loops.

3.1.3 RAFF File

The RAFF file is directly inspired from the previous notation and treelike robot arm structure. The comprehension of the file syntax and grammar is straightforward with the example shown in Figure 5.



```
RAFF MiniDelta

robot r1 {

    // path for the graphic files
    path “~GKG/Models/delta-4/”

    // here’s the base of robot
    base {    filename { “plateform.dxf” } }

    link 1 {            // first link of the robot
        parameters {
            theta       0.0         // in degrees
            r           0.0         // in mm
            d           1500.0      // in mm
            alpha       90.0        // in degrees
        }
        type 2          // type of joint 2=revolute
        pred 0          // predecessor = base (0)
        limits { -85.0 80.0 }
        filename { “bras.nff” }
    }

    link 2 {            // second link of the robot
        parameters {
            theta       211.21
            r           0.0
            d           2000.0
            alpah       0.0
        }
```

```
type 2
        pred 1          // attached to link 1
        limits { 185.0 355.0 }
    }

    // another link connected to the base
    link 6 {
        parameters {
            theta       0.0
            r           0.0
            d           1500.0
            alpha       90.0
            gamma       120.0     // some extra params
            epsil       0.0       // for this branch
        }
        type 2
        pred 0
        limits { -85.0 80.0 }
        filename { “bras.nff” }
    }

    // define the constraint for a closed loop
    constraint {
        link 5              // this two link must
        link 11             // be connected
    }

}   // end of robot definition
```

Figure 5: Example of a classic RAFF file

Multiple robot definitions are allowed in a single file. To be represented in the virtual world, each link of the robot can be represented by one or more graphic files. Theses 3D graphic files can come from various modelers as long as the graphic library used to build the virtual world can accept it. Often, a very realistic description of the robot’s parts is not required, since a simplified drawing can be more understandable.

In addition to the robot description, the RAFF file also currently contains the declaration of the potential interaction between the user and the robot parts. More precisely, all potential constraints between a link and the available sensors are declared, in the same way as a loop closure constraint, as shown in Figure 6. This means for example that one link can be moved with one sensor or another, or can be left free of constraints.

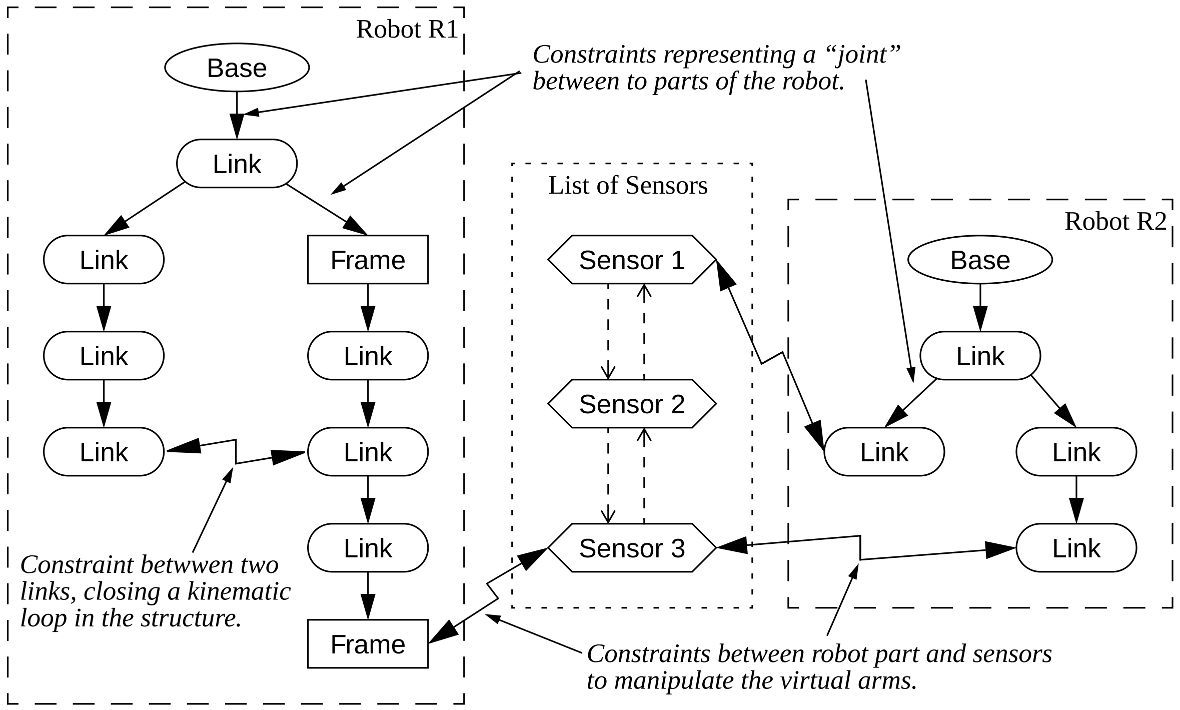
```
// declare a new sensor on serial port 2
// outside all other robot declaration !

sensor 1 {
      port "/dev/ttyd2"
      translation 3               // 3 dof in translation
      rotation 3                  // and 3 dof in rotation
}
```

```
// in a robot declaration
// define some potential constraints
constraint {
      link 5                      // link 5 must follow
      sensor ( 0 1 )              // either sensor 0 or
                        // sensor 1 (when 0, then the
                        // link is no more constrained)
}
```

Figure 6: Relations between the virtual robot and the user.

<u>3.2 Data Structure</u>

Regarding the previous definition of a robot structure, the more natural way to store it for processing is to use a graph. This graph contains different types of nodes and edges. This way we can store all the objects we need (see Figure 7), and it would be very easy to add new objects in the future. In addition the graph is oriented (the liaison between two objects can be unidirectional) to be as close as possible from the real structure: it is very easy to traverse the structure in order to update the coordinate transformations or to find the common predecessor of two links for example.



Robot R1
Base
Link
Link
Frame
Link
Link
Link
Link
Link
Frame
Constraints representing a "joint" between to parts of the robot.
List of Sensors
Sensor 1
Sensor 2
Sensor 3
Robot R2
Base
Link
Link
Link
Link
Constraint betwwen two links, closing a kinematic loop in the structure.
Constraints between robot part and sensors to manipulate the virtual arms.

Figure 7: Example of a structure containing different kinds of objects

<u>3.2.1 Nodes</u>

The minimum robot must contain at least one base. This is the fixed part of the robot arm. The base is necessary since it is the root of the robot structure, and it allows to place the robot somewhere in the space.

It is obvious that the most used object is the link object. It contains the Kleinfinger parameters as well as several transformations matrix for optimization.

An additional object was introduced for user convenience. It is a frame object which describes a coordinate system in function of a precedent object (base, link or frame). Such an object is useful when the coordinate transforms are not straightforward and can be described more easily with an intermediate coordinate system. For example, it is very interesting to use a frame for the description of the end-effector, since we can change its parameters without influencing the last link. In addition, it can be used to add some graphics files to the robotic system.

3.2.2. Edges

All the objects in a robot graph must be linked by edges. Each edge describes geometrical constraints between different objects, and then affects the behavior of the inverse kinematics algorithm. Currently, edges are of three types:
- Joint constraints, which connect two parts of the arm (links) with a revolute or prismatic joint.
- Loop constraints to close kinematics loops in the robot structure.
- Sensor constraints which can link a any object (base, link or frame) to a sensor. This indicates to the corresponding object to always try to follow the sensor movement. Each robot can have a set of constraints of this type which can be added or removed dynamically during the execution. This method provides a very practical way to control the robot since we can use different sensors to move different parts of the arm.

3.2.3. Sensors

A sensor is just an object to interface the user's input to the object in the virtual world. At each simulation loop, it reads the new input generated by the device, and sends it to the kinematic algorithm. Having sensors objects gives the following advantages: the core of the program remains independent of the input devices drivers, and it allows the interface to be truly flexible.

3.3. Constraints solver

The equations governing the robot's inverse kinematics are highly non-linear. However the variable themselves are non-linear, not their derivative. This is why we decided to work in velocities space rather than in the positions space. This leads to a linear transformation from velocities relative to a cartesian space into angular velocities for each joint of the robot. The matrix describing this linear transformation is called "manipulator jacobian".

The direct jacobian transforms the joint velocities into cartesian velocities. Its number of columns equals the number of degrees of freedom of the robot (see Figure 8), and its number of rows equals the number of degrees of freedom wanted in the operational space (usually six). The goal of the constraints solver is to build the inverse jacobian in order to find the joint velocities in function of user cartesian input velocities. This will be done through the numerical inversion of the direct manipulator jacobian.

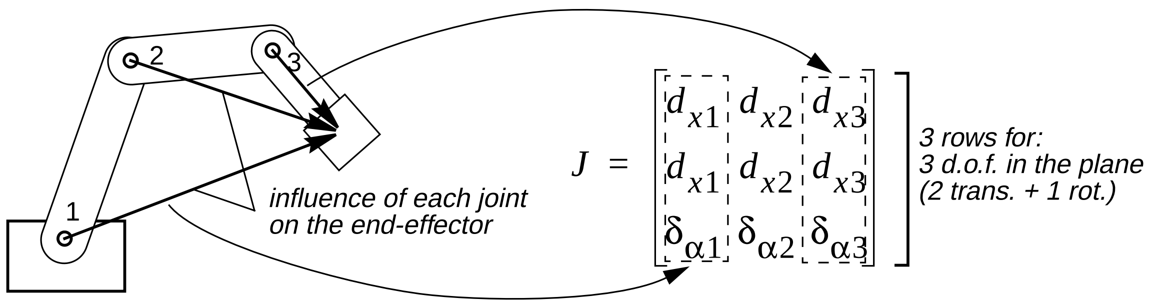

1
2
3
influence of each joint on the end-effector
$J = \begin{bmatrix} d_{x1} & d_{x2} & d_{x3} \\ d_{x1} & d_{x2} & d_{x3} \\ \delta_{\alpha1} & \delta_{\alpha2} & \delta_{\alpha3} \end{bmatrix}$
3 rows for:
3 d.o.f. in the plane
(2 trans. + 1 rot.)

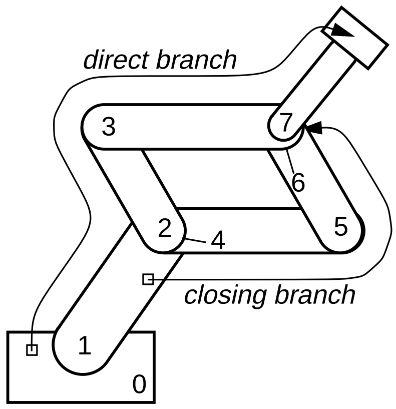
Figure 8: Construction of the Jacobian for a 3 d.o.f. manipulator moving in the plane

3.3.1 Jacobian generation

The construction of the direct jacobian is done with a general algorithm for all kind of tree like kinematic structures: each of its columns describes the influence of the considered joint onto the end-effector or the desired part of the robot in the general case. To allow closed loops or additional constraints in the robot structure, an "extended jacobian" is build with a set of additional rows representing constraints. In order to automatically generate the direct jacobian, the kinematic algorithm will traverse the robot graph generating as many sub-jacobians as the number of constrained branches in the manipulator structure (user or loop constraints). The extended jacobian will then contain all the sub-jacobians (see Figure 9).

3.3.2 Jacobian inversion

The direct jacobian allows the transformation from the angular variables into the cartesian variables. The inverse kinematics requires the inverse transform. Since it is a linear mapping, the inverse jacobian can be found by the inversion of the direct jacobian. However, since the manipulator can be either redundant or with less than 6 degrees of freedom, the jacobian matrix is non-square in most cases. This matrix inversion can lead to intrinsic numerical problems as well as non-sense for a real mechanical structure, requiring adequate methods for inversion. For example, the very efficient (in terms of computation-time) pseudo-inverse can be used and could give good results. However solutions given by inverse or pseudo inverse are unsatisfac-



$$J = \left[\begin{array}{ccccccc} d_{x1} & d_{x2} & d_{x3} & 0 & 0 & 0 & d_{x7} \\ d_{x1} & d_{x2} & d_{x3} & 0 & 0 & 0 & d_{x7} \\ d_{\alpha 1} & d_{\alpha 2} & d_{\alpha 3} & 0 & 0 & 0 & d_{\alpha 7} \\ 0 & -d_{x2} & -d_{x3} & d_{x4} & d_{x5} & d_{x6} & 0 \\ 0 & -d_{x2} & -d_{x3} & d_{x4} & d_{x5} & d_{x6} & 0 \\ 0 & -d_{\alpha 2} & -d_{\alpha 3} & d_{\alpha 4} & d_{\alpha 5} & d_{\alpha 6} & 0 \end{array}\right] \begin{array}{l} \left.\begin{array}{l} \\ \\ \\ \end{array}\right] \textit{end-effector constraint (user input)} \\ \left.\begin{array}{l} \\ \\ \\ \end{array}\right] \textit{closed-loop constraint (robot structure)} \end{array}$$

Figure 9: Structure of an extended jacobian for a hybride planar manipulator

tory near robot singularities (the rank of the jacobian change). Thus, the algorithm used in CINEGEN for the inversion of the jacobian is based on the singular value decomposition (SVD) of a matrix[18]. The SVD has been used both for kinematic and dynamic analysis, and real-time control of robotic manipulators. The SVD gives the following advantages:
- Robust algorithm which can deal with any kind of manipulators, serial parallel or hybrid, redundant or under-actuated.
- Possibility to use different approaches for singularities treatment such as the least square or dumped least square methods.
- Useful information about the approaches of a singularity allowing the use of visual aids to help the operator.

4. DISCUSSION

Several virtual manipulators were built with CINEGEN. Some representing their counterpart real robot, and others as a design exercise to study new manipulators (Figure 10 and Figure 11).

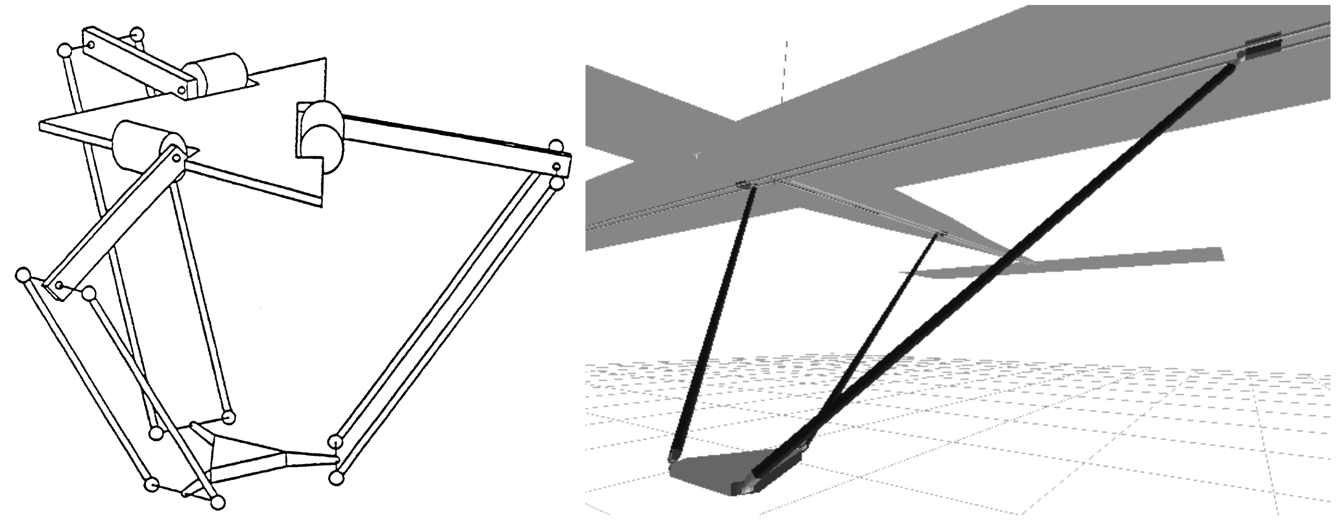


Figure 10: Original version of the delta4 robot and one of its virtual linear version

The current implementation of the program gives the expected performances about the real-time (we currently use Silicon Graphics workstations). The global loop of the simulation, including the sensor reading, the inverse kinematic calculations and the rendering, can be done more than 30 times per second for the manipulators presented in this paper (it should be noted that the rendering is almost completely done by the graphic card, thus does not consume any CPU time).

According to our results, the RAFF file has proven its ability to entirely define any kind of manipulators in a very simple way. It is possible to describe a new manipulator and then simulate it in less than one hour. In addition, the RAFF file is able to describe the relation of the manipulator with the user through multiple 3D sensors.

Compared to the classical approaches used in robotics to drive manipulator arms, the virtual interface really improves the manipulability of the arm: it is very intuitive to guide an arm by its end-effector, since it corresponds to the natural way in which we would execute the task. In addition, the user can easily switch from one coordinate frame to another, what provides a real flexibility in the command.

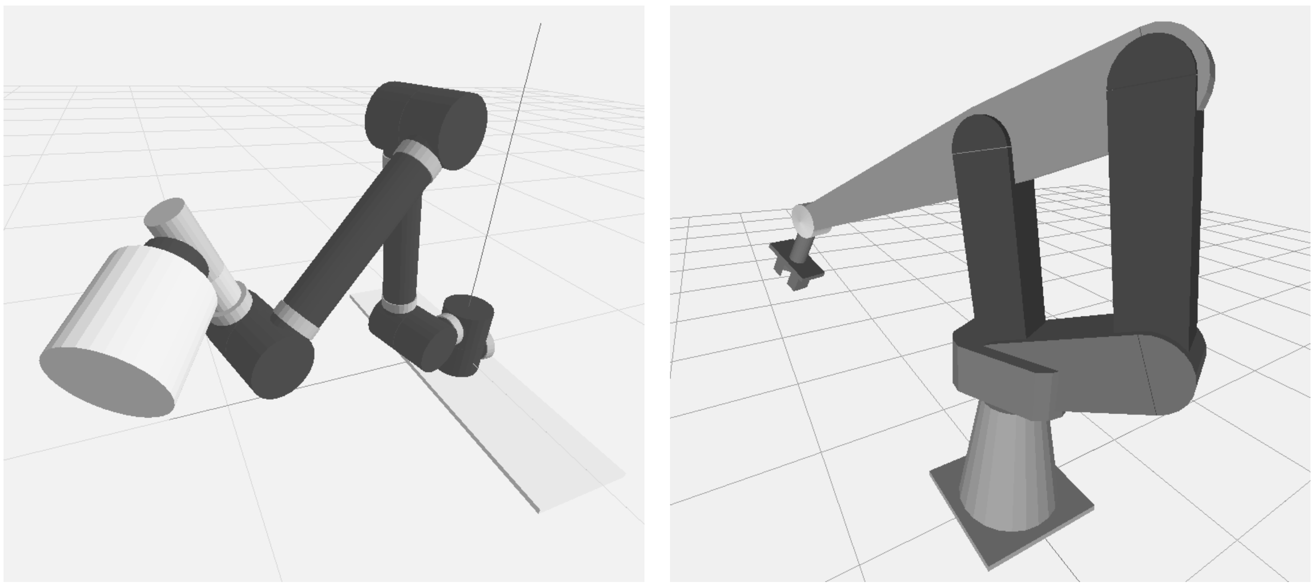


Figure 11: Example of a serial 5 dof manipulator and an hydride manipulator in the virtual environement

The possibility to drive in the same way a robot arm with less or more degrees of freedom than the operational task (usually 6) gives the user a very coherent interface. For under-actuated arms, the user sees immediately when he can't perform all the movements he wants, and can then adapt the task. The redundant manipulator arms are easily maneuverable as well, since the kinematic algorithm finds an acceptable configuration through the "least square" solution. However, additional behavior such as automatic obstacle avoidance or other optimization for a redundant arm have not been implemented yet.

The performance of the SVD for the kinematic algorithm provides powerful results for both real-time behavior and kinematic studies. The characteristics of the SVD could help a lot in the measurement of the dexterity and the singularities treatment. In addition to automatic singularities avoidance, it is possible for example to display in advance the singular configuration in which the robot will go, and then provide the user manual tools to adapt the behavior of the robot.

5. CONCLUSION

This paper presents an overview of the CINEGEN project, which is designed as a new virtual environment based interface for robot manipulator's design, control and visualization. We have seen that choosing the adapted tools for modeling and computing allows to obtain an automatic kinematics generator for serial and parallel manipulators. The performance of the system allows a real time rendering which lets the user pilot the robot in a very intuitive way.

The results already obtained show that CINEGEN is able to answer to the two common problems encountered by the IMT and the IMG:

- Minimize the time to develop a new interface when a mission on the field is planned with a robot arm which exists.
- Optimize testing of the kinematic of a robot arm during its design, according to a specified task, before the real construction of the arm.

Further developments will focus on improving the interface, not only by adding more automated tasks, but by inserting the user in the control loop when special behaviors are encountered. This should be achieved by keeping in mind that the robot interface is first a tool for humans.

6. ACKNOWLEDGMENTS

This program is supported by the Swiss National Science Foundation (grant # 2100-040759.9401).

The authors would like to thank Professor R. Clavel, head of the IMT (Swiss Federal Institute of Technology) for encouraging research in that field.

The first author also would like to thank B. Hine, ph.D. and M. Sims, ph.D. past and present group leaders of the IMG (NASA Ames Research Center) for inviting him to work at NASA Ames Research Center in the context of his ph.D.

7. REFERENCES

1. E. Natonek, L. Flückiger, T. Zimmerman and C. Baur, “Virtual Reality: an intuitive approach to robotics”, *SPIE Telemanipulator and Telepresence Technologies*, Vol. 2351, pp. 260-269, Boston, Oct-Nov 1994.

2. E. Natonek, T. Zimmerman and L. Flückiger, “Model based vision feedback for virtual reality robotics environments”, IEEE *Virtual Reality Annual Symposium*, pp. 110-117, March 1995.

3. A. Sulzmann and C. Baur, “A virtual reality based interface for micro-telemanipulation”, *ICAT’94*, pp.255-262, Tokyo, July 1994.

4. L. Piguet, T. Fong, B. Hine, P. Hontalas and E. Nygren, “VEVI: A virtual reality tool for robotic planetary explorations”, *Virtual Reality World*, vol. pp. 263-274, Stuttgart, Germany, February 1995.

5. B. Hine, P. Hontalas, T. Fong, L. Piguet, E. Nygren and A. Kline, “VEVI: A virtual environment teleoperation interface for planetary explorations”, *SAE 25th International Conference on Environmental Systems*, San Diego, CA, July 1995.

6. B. Roth, “Computations in Kinematics”, *in Computational Kinematics*, pp. 3-14, Dordrecht, The Netherlands, 1993.

7. B. Roth, “Computationnal advances in robot kinematics”, *in Advances in Robot Kinematics and Computational Geometry*, pp. 7-16, Dordrecht, 1994.

8. M. Raghavan and B. Roth, “A general solution for the inverse kinematics of all series chains”, *in RoManSy 8*, pp. 24-31, Warsaw, Nowowiejska, 1990.

9. L. W. Tsai and A. P. Morgan, “Solving the kinematics of the most general six- and five-degree of freedom manipulators by continuation methods”, *Journal of Mechanisms, Transmissions, and Automation in Design*, vol. 107, pp. 189-200, 1985.

10. C. W. Wampler, A. P. Morgan and A. J. Sommese, “Numerical continuation methods for solving polynomial systems arising in kinematics”, *Journal of Mechanical Design*, vol. 112, pp. 59-68, 1990.

11. J. Hollman and L. Langemyr, “Algorithms for Non-Linear Algebraic Constraints”, *in Constraint Logic Programming. Selected Research*, pp. 113-131, Cambridge, 1993.

12. Y. Kuroe, Y. Nakai and T. Mori, “A new neural network learning of inverse kinematics of robot manipulator”, *Internatinal conference on neural networks*, vol. V, pp. 2819-2824, Orlondo, Florida, June 27- June 29, 1994.

13. G. Wu and J. Wang, “A recurent neural network for manipulator inverse kinematics computation”, *Internatinal conference on neural networks*, vol. V, pp. 2715-2720, Orlondo, Florida, June 27- June 29, 1994.

14. J. Denavit and R. Hartenberg, “A kinematic notation for lower pair mechanism based on matrices”, *in Journal of Applied Mechanics*, pp. 215-221, 1955.

15. R. P. Paul, “Robot manipulators: mathematics, programming, and control. The Computer Control of Robot Manipulators”, The MIT Press, Cambridge, 1983.

16. P. N. Sheth and J. J. Uiker, “A generalized symbolic notation for mechanisms”, *Journal of Engineering for Industry*, vol. 93, pp. 102-112, 1971.

17. J.F. Kleinfinger, “Modélisation dynamique de robots à chaine cinématique simple, arborescente ou fermée en vue de leur commande”, *Thesis*, Nantes, 1986. *(in french)*

see also: W. Khalil and J.-F. Kleinfinger, “A new geometric notation for open and closed-loop robots”, *IEEE conf. on Robotics and Automation*, San Francisco, 1986.

18. A. A. Maciejewski and C. A. Klein, “The singular value decomposition: computation and applications to robotics”, *Robotics Research*, vol. 8, pp. 63-79, 1989.