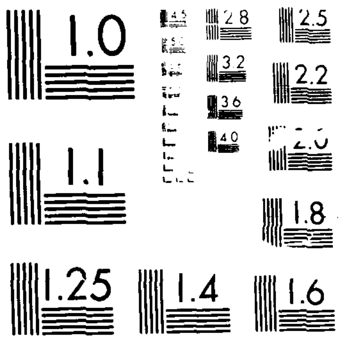
[illegible]

MULTI-DEGREE-OF-FREEDOM MOTIONS(U) CALIFORNIA UNIV DAVIS DEPT OF MECHANICAL ENGINEERING B RAVANI

UNCLASSIFIED 30 OCT 89 ARO-26122.2-EG DAAL03-88-K-0206 F/G 12/9 NL


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

6c. ADDRESS (City, State, and ZIP Code) Davis, CA 95616		7b. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U. S. Army Research Office	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAL03-88-K-0206		
8c. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211		10. SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Classification and Simulation of Single and Multi-Degree-of Freedom Motions (Unclassified)

12. PERSONAL AUTHOR(S)
Bahram Ravani

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 88-10-1 TO 89-9-31	89-10-30	

16. SUPPLEMENTARY NOTATION The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Robotics, Dynamics, Simulation, CAD

CLASSIFICATION AND SIMULATION OF SINGLE AND MULTI-DEGREE-OF-FREEDOM MOTIONS

A FINAL REPORT

BY

BAHRAM RAVANI
ASSOCIATE PROFESSOR

OCTOBER 27, 1989

US ARMY RESEARCH OFFICE
GRANT NUMBER DAAL-03-88-K-0206
GRANT DURATION: SEPTEMBER 1988 - SEPTEMBER 31, 1989
GRANT AMOUNT: $19,100

DEPARTMENT OF MECHANICAL ENGINEERING
UNIVERSITY OF CALIFORNIA-DAVIS
DAVIS, CA 95616
(916) 752-4986

INTRODUCTION

This research program has been the final extension of a previous research grant that was completed on August 31, 1988 and was reported in the final report dated September 12, 1988. The work described here is the final part of the effort covering the period from September 1988 to September 31, 1989.

The work since September of 1988 has resulted in the following developments:

- scientific methods for generation of robot trajectories that can deliver maximum impulse at the end-point of the trajectory.
- design of the world modeling system RWORLD (Robot WORLD) for CAD Based robot programming and simulation.
- final development and completion of the computer software OPTLOAD for calculation of load optimal robot trajectories.
- initial work in the development of a theoretical basis for world model calibration based on features.

The results of the above developments have been or are being published in the following technical publications:

A. PUBLISHED

1. Ravani, B., "World Modeling for CAD Based Robot Programming and Simulation," Int'l Journal of Robotics and Automation, Vol. 4, No. 2, 1989, pp. 96-105.

B. PLANNED PUBLICATIONS

1. Wang, L.T. and B. Ravani, "Maximum Impulse Robot Trajectories," Technical Note, to be submitted to the ASME Journal of Dynamic Systems, Measurements and Control.
2. Ravani, B., S. Serrien, and H. Van Brussel, "World Model Calibration in Robot Simulation," to be submitted to Journal of Manufacturing Systems.

The available set of these publications (A.1) or their theoretical basis (B.2) is given in Appendix A. These publications are in addition to the eight technical publications described in the Final Report of 12 September 1988.

SUMMARY OF SCIENTIFIC PROGRESS AND RESULTS

A method has been developed that would allow generation of robot trajectories that can deliver maximum impulse at a specified end-point of their trajectory. Since impulse can be related to the dynamic payload, the method for generation of load optimal trajectories developed earlier has been modified to handle the problem. The resulting scheme has applications in military applications of robotics involving automatic loading of weapon systems. In such a situation proper engagement of the ammunition in certain weapons require an impulsive motion from the robot.

In the area of CAD based simulation, the initial design of a world modeling system has been completed. The system referred to as RWORLD (Robot WORLD) is designed using our experience base from the development of the off-line robot simulation system STAR. RWORLD, however, advances the state of the art in CAD based robot programming and simulation in several areas. The system can support interfaces to the actual robot workcell environment allowing for calibration of the workcell for inaccuracies. The system models geometrical, kinematical, relational and physical properties of the world, allowing for geometrical and spatial reasoning as well as reasoning about the mechanics of manipulation. It also supports simulation of several sensory functions and multiple arm coordinated control.

The initial design of the RWORLD has also been modified to be able to perform world model calibration based on geometric features. This is the first time that world model calibration is addressed from a feature level point of view. Furthermore, the addition of a feature primitive to the data structure of RWORLD has opened up a new direction of research in feature based programming and simulation. This together with expansion of RWORLD is part of the basis for our newly funded ARO project.

SCIENTIFIC PERSONNEL

The following personnel were supported on a partial basis during different phases of this project over the last year:

a. Principal Investigator: Professor B. Ravani

b. Graduate Students:

1. S. Serrien
2. C. Choi

APPENDIX A

RELATED PUBLICATIONS AND TECHNICAL INFORMATION

International Journal of Robotics and Automation, Vol. 4, No. 2, 1989

WORLD MODELING FOR CAD-BASED ROBOT PROGRAMMING AND SIMULATION

Bahram Ravani (*)

Abstract

The design of a world modeling system for CAD-based robot programming and simulation is presented. The system can support interfaces to actual robot workcell environments, allowing for calibration of the workcell model for inaccuracies and its use for robot control. The system models geometrical, spatial, relational, and physical properties of the world, allowing for geometrical and spatial reasoning as well as reasoning about the mechanics of manipulation. It supports simulation of several sensory functions and multiple-arm coordinated control. It also supports representation of assemblies and aggregation of multiple devices. The system design presented is the basis of a world modeling system for model-based robot task planning, simulation, and control currently under development.

Key Words

Geometric Modeling, Robotic Programming, CAD/CAM

1. Introduction

World modeling is the process of providing a computer representation of a workcell in robotic applications. The world model provides the necessary knowledge of the robot workcell for a task planner to generate appropriate robot motion control commands. The task planner accesses the world model when converting implicit or explicit motion commands into trajectory directives, and updates the world model when the state of the workcell configuration changes. During robot task execution, for example, parts are grasped, repositioned, and released or assemblies are created. All this must be reflected within the world model.

In CAD-based robot programming, the world model includes CAD models of the robot and its workcell environment. If the programmer performs reasoning for task-planning purposes by interacting with a computer graphics (CAD) display of the robot and its environment (generated from the data in the world model), the programming or the task-planning process is called interactive or off-line (see for example, [1,2]). On the other hand, if a computer program (using the data in the world model) performs the reasoning for task planning, the process is called automatic robot task planning and programming (see for example, [3,4]).

† This paper was first published in NATO ASI Series F, Vol. 50, *CAD-Based Programming for Sensor Robots*, ed. B. Ravani, Springer Verlag, 1988.

(*) Department of Mechanical Engineering, University of California - Davis, Davis, CA 95616 USA

This paper describes the design of the world modeling system RWORLD (Robot WORLD), which is the basis of a CAD based (off-line) robot programming and simulation environment presently under development in our laboratory. In contrast to its predecessor off-line planning system STAR ([5]), RWORLD has real time interfaces to the actual robot workcell both for correcting inaccuracies in the world model as well as for model based robot control. The system has several advanced capabilities providing for an inactive environment for robot task planning, simulation, and control. It models geometrical (CAD), relational, and physical (kinematic and dynamic) properties of the world. These allow for geometrical and spatial reasoning as well as reasoning about the mechanic of manipulation. RWORLD is also structured to support parallelism (parallel execution) for simulation of multiple cooperative robot arms operating at the same time. Furthermore, it allows definition of aggregation and assemblies to handle simulation of tool changes and creation of component assemblies. It also uses sensory processing for world model calibration eliminating or reducing modeling errors. Finally, the system provides functions for geometric simulation of sensory interactions allowing emulation of error recovery during interactive task planning.

Each of the foregoing features is described in more detail in subsequent sections.

2. Modeling of Robotic Workcell Environments

In off-line planning and simulation, the simulator requires an appropriate model of the robot world. In robotic control, the planner also accesses the same world model. The world model should include a geometrical as well as a physical description of the world. It should also describe the relationships between different entities in the world. The geometrical description is needed for geometrical reasoning and the physical description is needed for reasoning about the mechanics of manipulation. The relational description allows for spatial and temporal reasoning and planning.

In a complex application, a manipulator may use several different tools and end-effectors, interact with other equipment or manipulators, and use sensory data and process parts to create component assemblies or products. The tools or end-effectors used may be other complex manipulators such as dexterous mechanical hands. The parts may also be subassemblies of other parts. This means that robot workcells consist in hierarchical or non-hierarchical aggregations of several distinct elements such as parts, equipment, and sensors, each having different physical representations. A rigid part, for example, does not have any internal kinematic representation, while a manipulator does have an internal kinematic representation due to the relative freedom between its links. For this reason, RWORLD uses a multi-primitive representation of the workcell environment at an abstract level. These abstract primitives are devices, rigid objects (objects for short), frames, and sensors. There are also a set of affixment descriptors representing the relation between these primitives.

A device is an entity having one or more internal degrees of freedom. A device can affect objects within its workspace by imparting motion to them, exerting a force on them, or processing them. Examples of devices are robots, machine tools, conveyor belts, end-effectors, and flexible fixtures. A rigid object is an entity with no internal degree of freedom but that is capable of being moved with six degrees of freedom in the workcell environment. Examples of objects are workpieces and tables. The two primitives of devices and objects are defined in a hierarchical fashion such that objects can be accessed by devices but devices cannot be accessed by objects. This is because objects can be processed or manipulated by devices but devices cannot be manipulated or processed by objects. Frames are the third primitive in the world model: they are used to mark a location, i.e., a position and orientation in space. Such frames are referred to as *natural frames*. In addition to their use as one of the basic elements of the world model, frame primitives are also used to mark feature locations on objects or devices. Such frames are referred to as ***auxiliary frames***. Frames indicating target locations in space can also be considered as auxiliary frames for the free space. Sensors are the last primitive within the world model. A sensor primitive provides a functional representation for sensory interactions for simulation purposes. RWORLD design facilitates emulation of touch, tactile, proximity, and presence-sensing functions.

The sensor primitive has the same hierarchical level as the object primitive within the data structure: it can therefore be attached to devices and objects. Frames are in the lowest level of hierarchy. They can be attached to themselves, objects or sensors, and devices. Fig. 1 illustrates the hierarchical organization of these four primitives.

The four primitives of frames, objects, sensors, and devices form the basis for spatial, geometrical, relational, and physical representation of the world. RWORLD supports multiple representation of these primitives at geometrical and physical levels to speed up the computations necessary at different levels of robot task planning, simulation, and programming.

3. Spatial and Geometrical Representation of the World

The spatial and geometrical representation of the robot workcell can be described by assigning certain ownership properties for the four abstract primitives of frames, objects, sensors, and devices. A frame primitive is then represented by a name and will own a homogeneous transformation matrix and a CAD descriptor. The transformation matrix describes the spatial relationship between the frame and a global coordinate frame in the workcell, and the CAD descriptor provides a geometric or shape representation of the frame. This representation is in the form of three mutually orthogonal axes. A frame primitive can be instantiated based on its spatial relationship descriptor (a homogeneous transformation matrix). Fig. 2 illustrates the structure of the frame primitive. Instantiation of a frame primitive allows definition of relative frames.

A rigid object primitive is represented by a name a natural frame representing the spatial relationship of the object relative to another frame, and a set of properties such as mass, inertia, and center of mass location. The center of mass location is specified with respect to the natural frame of the object. The set of properties can also include other characteristics of the objects such as material properties needed in modeling interactions between objects or surface reflectance and other properties needed for vision-controlled manipulation. The information on mass, inertia, and center of mass location allow for multiple physical representation of an object in terms of both a rigid body model as well as a more simplified lumped mass model. This dual representation simplifies some of the computations necessary when reasoning about or simulating the mechanics of manipulation.

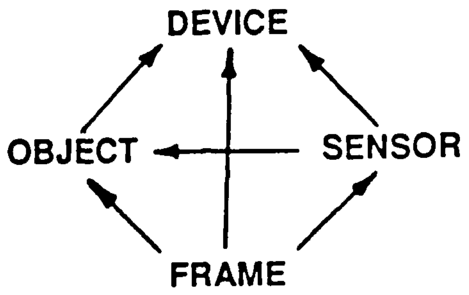


Figure 1. The hierarchical organization of RWORLD primitives

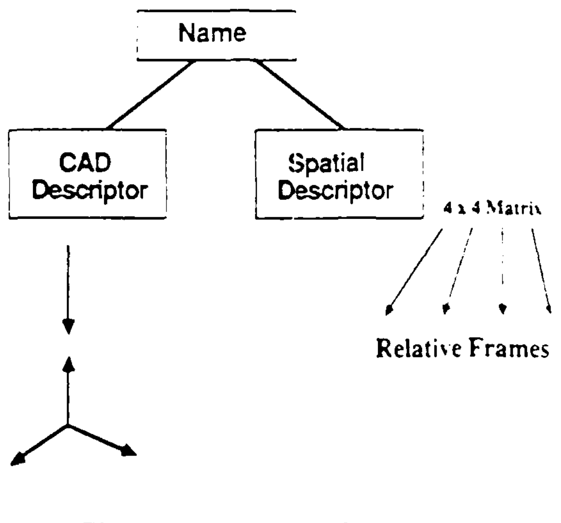


Figure 2. The structure of the frame primitive and its instances.

An object primitive (Fig. 3) owns a CAD descriptor providing its geometric (shape) representation and possibly one or more auxiliary frames indicating feature locations on the object. RWORLD uses a polyhedral solid modeling system for its geometric or CAD representation. Furthermore, it supports two instances of CAD description, one in terms of the polyhedral model and one in terms of the more simplified bounding parallelpiped (bounding box) or prismatic representation.

An object can also be instantiated based on the homogeneous transformation matrix associated with its natural frame. Different instances of an object, in this context, then represent different locations of the same object in the workspace or different references to the same object in terms of relative frames. These frames are usually represented either with respect to the natural frame of the object or with respect to other auxiliary frames representing the more basic features of the same object.

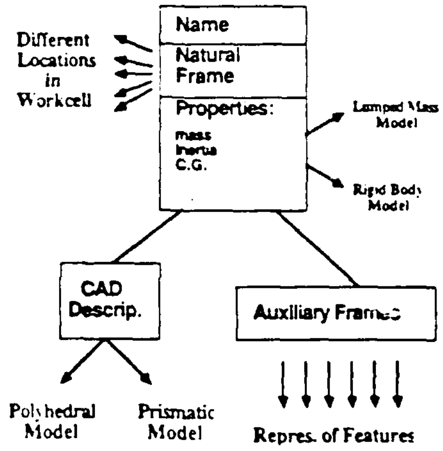


Figure 3. The object primitive and its instances.

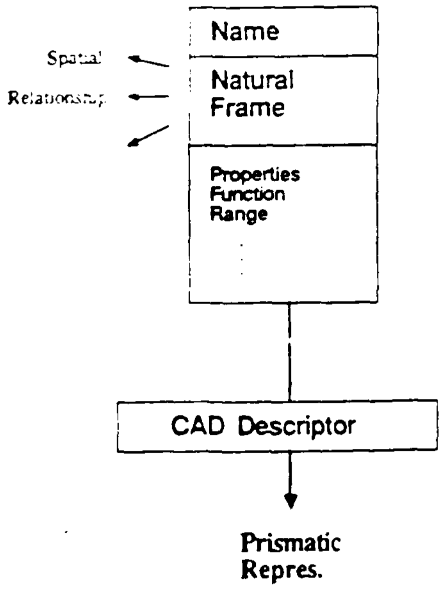


Figure 4. The sensor primitive and its instances.

A sensor primitive (Fig. 4) is represented by a name, a type descriptor, and a natural frame; it owns a geometric CAD descriptor. The type descriptor indicates whether the primitive represents a touch, a tactile, a proximity, or a presence sensor. The natural frame represents the spatial relationship of the sensor in the workcell. Within RWORLD, sensors are modeled geometrically in terms of prismatic objects only. This means that there is only one instance of the CAD descriptor for a sensor.

The representation of the device primitive is more complicated than the other primitives, since devices have internal degrees of freedom and are programmable or require control statements. A device primitive (Fig. 5) is represented by a name, a natural frame, a mobility descriptor, and a set of joints and links. It owns a set of motion programming commands and a Tool Center Point (TCP). The natural frame is used to represent the spatial relationship between the base of the device and another frame within the workcell. The mobility descriptor specifies the number of internal degrees of freedom of the device. The joints and links are lower level primitives residing subordinate to a device primitive.

A joint primitive of a device is represented by a name, the joint number within the device (the present design of RWORLD can only model devices made of serial kinematic chains), and a set of joint trajectory limits. These limits indicate the bounds on joint displacement, velocity, and acceleration limits as well as limits on the torque (or force) provided at the joint by the actuators. Such information is needed to reason about the mechanics of joint motion. The joint primitive owns a set of joint trajectories consisting of joint displacement, velocity, and acceleration. It also owns a CAD descriptor for computer graphic display purposes.

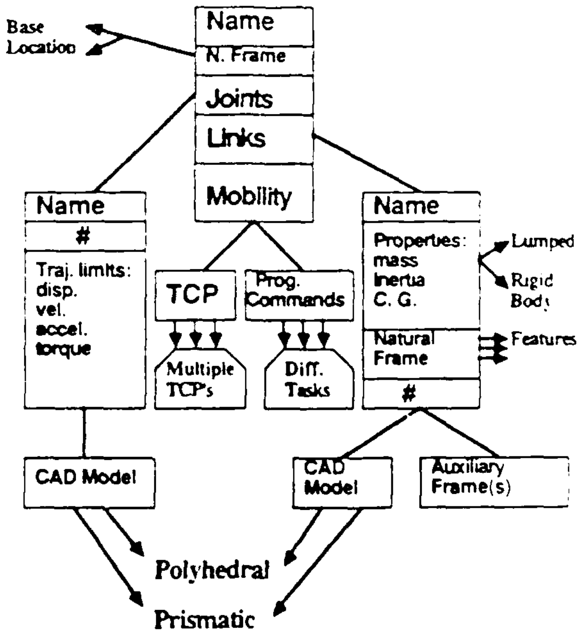


Figure 5. The device primitive and its instances

A link primitive of a device contains a link name, the link number within the device, a natural frame, and a set of link properties, including link mass, inertia, and center of mass location. The primitive owns a geometric CAD descriptor and possibly one or more auxiliary frames representing features on the link. The geometric representation of a link also has two instances, one in terms of a polyhedral model and one in terms of a prismatic approximation of the polyhedral representation.

The TCP of a device is an auxiliary frame representing a coordinate system on the last link of the robot, in terms of which the motion program of the robot is specified. The set of motion-programming comands is specified in terms of this frame, either by an explicit robot-programming language or by synthesis from implicit task-level specification of a robot operation. Instantiation of the coordinate transformation of the TCP in terms of other feature frames of the device allows definition of multiple tool center points. The set of motion-programming commands specifies the manipulation function of a device in carrying out a task. Different instances of this set represent different tasks that are to be completed by a device.

The specifications of the four primitives of frames, objects, sensors, and devices provide for a complete representation of the geometry and spatial relationships between the elements of a robotic workcell environment.

4. Physical Representation of the World

The physical representation of a robot workcell (world) should emulate kinematics as well as dynamics of objects and devices in the workcell environment. It should also include models for gravity, friction between surfaces, and interactions between various elements of the workcell. This is so because robotic manipulation involves imparting motion to objects as well as applying forces and moments to them.

The representation of mass, inertia properties, and center of mass location for objects (as discussed in the previous section) facilitates physical modeling of objects using Newton's laws of motion. The development of physical (kinematic and dynamic) models of devices, however is slightly more involved due to the relative degrees of freedom between their links.

A key consideration for physical modeling of devices is the computational efficiency of the resulting formulation necessary for simulation purposes. The multi-link nature of a device such as a robot suggests the use of a recursive computational formalism. RWORLD automatically generates kinematic and dynamic models for devices in a form that would allow computations with forward or backward recursions. This provides enough flexibility in simulating physical models of devices in real time.

Fig. 6 is a skeleton diagram of a six-degree-of-freedom robot manipulator with the natural frames of its links numbered from one to six. In this figure, the seventh coordinate frame represents the TCP of the robot. The kinematic equations for this device describe the spatial relationship between the seventh and the first coordinate systems in terms of the joint variables of the robot. These equations can be derived recursively by considering the spatial relationship between the coordinate systems of two adjacent links.

Fig. 7 shows the ith and the $(i+1)$th coordinate systems connected through the ith link. Let $(\mathbf{x}_j \mathbf{y}_j \mathbf{z}_j)(j = 1, i+1)$ be s set of three orthogonal unit vectors coincident, respectively, with the axes of the ith and the $(i+1)$th coordinate systems. Since these unit vectors form a dextral set, two of them completely define the orientation of the corresponding coordinate system. From the geometry in Fig. 7, one can write [6]:

$$\mathbf{x}_{i+1} = \mathbf{x}_i \cos\theta_i + \mathbf{y}_i \sin\theta_i \quad (1)$$

and

$$\mathbf{z}_{i+1} = \mathbf{z}_i \cos\alpha_i + (\mathbf{x}_{i+1} \mathbf{X}\, \mathbf{z}_i) \sin\alpha_i \quad (2)$$

Substituting from (1) into (2), this last equation simplifies to

$$\mathbf{z}_{i+1} = \mathbf{z}_i \cos\alpha_i + (\mathbf{x}_i \sin\theta_i - \mathbf{y}_i \cos\theta_i) \sin\alpha_i \quad (3)$$

Equations 1 and 3 form a set of two forward recursive formulas for calculating the orientation of the $(i+1)$th coordinate system in terms of that of the ith coordinate system.

Similarly, a pair of backward recursive formulas can be derived as

$$\mathbf{z}_i = \mathbf{z}_{i+1} \cos\alpha_i + \mathbf{y}_{i+1} \sin\alpha_i \quad (4)$$

$$\mathbf{x}_i = \mathbf{x}_{i+1} \cos\theta_i + (\mathbf{z}_{i+1} \sin\alpha_i - \mathbf{y}_{i+1} \cos\alpha_i) \sin\theta_i \quad (5)$$

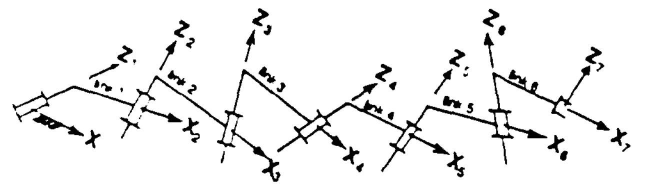


Figure 6. Skeleton diagram of a robot with natural coordinate systems of its various links.

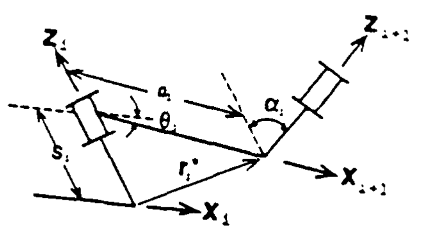


Figure 7. Two neighboring joints of a robot.

Repeated use of these last two equations allow computation of the orientation of the TCP coordinate system with respect to the base or first coordinate system. The set of two equations (1) and (3) or (4) and (5) each require 17 multiplications and three additions plus four trigonometric evaluations for their computations. The forward recursive formulas perform the computations from the ith coordinate system to the $(n-1)$th coordinate system (n is the number of degrees of freedom of the device). The backward formulas, however, compute from the ith coordinate system to the base coordinate system.

For most commercially available manipulators, the link twist angles, α_i's, are either 0° or 90°. The resulting simplified equations are:

For forward recursion:
With $\alpha_i = 0°$:

$$\mathbf{x}_{i+1} = \mathbf{x}_i \cos\theta_i + \mathbf{y}_i \sin\theta_i ; \ \mathbf{z}_{i+1} = \mathbf{z}_i \tag{6}$$

With $\alpha_i = 90°$:

$$\mathbf{x}_{i+1} = \mathbf{x}_i \cos\theta_i + \mathbf{y}_i \sin\theta_i \mathbf{z}_{i+1} = \mathbf{x}_i \sin\theta_i - \mathbf{y}_i \cos\theta_i \tag{7}$$

For backward recursion:
With $\alpha_i = 0°$:

$$\mathbf{x}_i = \mathbf{x}_{i+1} \cos\theta_i = \mathbf{y}_{i+1} \sin\theta_i ; \ \mathbf{z}_i = \mathbf{z}_{i+1} \tag{8}$$

With $\alpha_i = 90°$:

$$\mathbf{x}_i = \mathbf{x}_{i+1} \cos\theta_i + \mathbf{z}_{i+1} \sin\theta_i ; \ \mathbf{z}_i = \mathbf{y}_{i+1} \tag{9}$$

The spatial relationship between the $(i-1)$th and ith coordinate system also requires computation of position of the origin of one coordinate system with respect to the other. The position of the origin of the $(i-1)$th coordinate system with respect to the ith coordinate system is specified by a position vector $\mathbf{r}_i^*$ (Fig. 7). This vector can be written in terms of the link length a_i and the joint offset s_i as

$$\mathbf{r}_i^* = s_i \mathbf{z}_i + a_i \mathbf{x}_{i+1} \tag{10}$$

Now if ${}^j\mathbf{r}_i$ represent the position vector from the origin of the ith coordinate system to that of the jth coordinate system, then the forward recursive formula for the computation of the position of a coordinate system becomes:

$${}^j\mathbf{r}_{i+1} = {}^j\mathbf{r}_i = \mathbf{r}_i^* \tag{11}$$

with ${}^j\mathbf{r}_i = 0$
and $\mathbf{r}_i^*$ is given in equation (10). For backward recursion the corresponding equation is

$${}^j\mathbf{r}_i = {}^j\mathbf{r}_{i+1} + \mathbf{r}_i^* \tag{12}$$

Equations 11 and 12 each require six multiplications and two additions.

Repeated use of the recursive formulas for position and orientation allows computation of full kinematics of a robot manipulator or spatial relationships between two coordinate frames related through a set of other auxiliary or natural frames.

In addition to the kinematic equations, reasoning about the mechanics of manipulation requires modeling of quasi-static force/torque transmission characteristics of a robot as well as computation of inverse dynamics of the manipulator. Such information is also needed for control purposes if the robot is under force and/or dynamic control.

Even when the robot is not under dynamic control, the inverse dynamic computations are needed to reason about the TCP path traversal speed and robot motion cycle times that will avoid overloading of the [illegible]. Such computations are also necessary for load capacity determination [7] and planning of optimal trajectories. The computation of quasi-static force transmission characteristics of a manipulator allows reasoning about the forces (torques) that are to be applied to workpieces in force control applications.

RWORLD uses an algorithm that computes the inverse dynamics and the quasi-static force transmission characteristics of the manipulator simultaneously. The latter is determined by the computation of the so-called Jacobian matrix for the manipulator [6]. The inverse dynamics allow computation of joint torques (forces) necessary to achieve a desired TCP velocity and acceleration. The Jacobian matrix allows computation of the joint torques (or forces) necessary to apply a desired set of forces and/or torques at the TCP frame. The computational scheme used in RWORLD uses the backward kinematic equations together with the Newton-Euler formulation of manipulator dynamics.

The total inertia force and inertia moment exerted on the ith link of a manipulator can be computed, respectively, from (Fig. 8)

$$\mathbf{F}_i = m_i \mathbf{V}_{ci} \tag{13}$$

$$\mathbf{N}_i = [\text{illegible}]\, \mathbf{w}_i + [\text{illegible}]\, \mathbf{w}_i + [\text{illegible}]\, \mathbf{w}_i \tag{14}$$

where ${}^{i+1}\mathbf{w}$ and ${}^{i+1}\dot{\mathbf{w}}$ are respectively the angular velocity and angular acceleration of link i with respect to the $(i+1)$th moving coordinate system; [illegible] is the 3 × 3

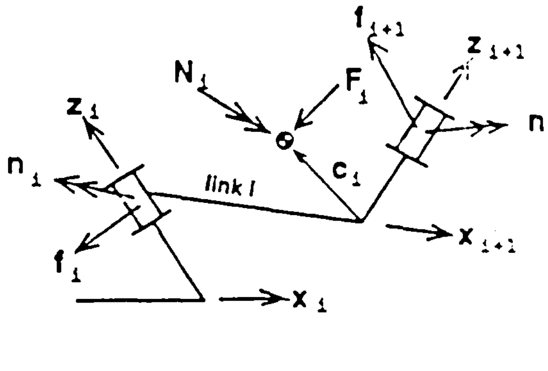


Figure 8. Forces and moments exerted on the ith link

inertia matrix of link i evaluated about its mass center and referenced to its natural frame (i.e., the $(i+1)$th coordinate system). The angular velocity, ${}^{i+1}\mathbf{w}_i$, and the angular acceleration, ${}^{i+1}\dot{\mathbf{w}}_i$ are computed from

$$ {}^{i+1}\mathbf{w}_i = [{}^{n+1}R_{i+1}]^T \mathbf{w}_i;\ {}^{i+1}\dot{\mathbf{w}}_i = [{}^{n+1}R_{i+1}]^T \dot{\mathbf{w}}_i \qquad (15) $$

where

$$ [{}^{n+1}R_{i+1}] = [\mathbf{x}_{i+1}\mathbf{y}_{i+1}\mathbf{z}_{i+1}] $$

and for a rotational joint

$$ \mathbf{w}_i = \mathbf{w}_{i-1} + \mathbf{z}_i\dot{q}_i;\ \dot{\mathbf{w}}_i = \dot{\mathbf{w}}_{i-1} + \mathbf{w}_{i-1} \mathrm{x}\, \mathbf{z}_i\dot{q}_i + \mathbf{z}_i\ddot{q}_i \quad (16) $$

For a translational joint, these equations simplfy to where q_i's are the joint positions (for revolute joints $q_i = \theta_i$).

$$ \mathbf{w}_i = \mathbf{w}_{i-1};\ \dot{\mathbf{w}}_i = \dot{\mathbf{w}}_{i-1} \qquad (17) $$

The linear acceleration of the mass center of link i (in Equation 13) is computed from

$$ \dot{\mathbf{V}}_{ci} = \dot{\mathbf{w}}_i \times \mathbf{c}_i + \mathbf{w}_i \times (\mathbf{w}_i \times \mathbf{c}_i) + \mathbf{V}_i \qquad (18) $$

where for a rotational joint

$$ \dot{\mathbf{V}}_i = \dot{\mathbf{V}}_{i-1} + \mathbf{w}_i \times (\mathbf{w}_i \times \mathbf{r}_i^*) + \dot{\mathbf{w}}_i \times \mathbf{r}_i^* $$

but for a translational joint

$$ \dot{\mathbf{V}}_i = \dot{\mathbf{V}}_{i-1} + \dot{\mathbf{w}}_i \times \mathbf{r}_i^* + \mathbf{w}_i \times (\mathbf{w}_i \times \mathbf{r}_i^*) + 2\mathbf{w}_i \times \mathbf{z}_i\dot{\mathbf{q}}_i + \mathbf{z}_i\ddot{q}_i $$

the relationships and the constraints existing between various elements of the world should also be properly represented and maintained. RWORLD uses affixment descriptors and ownership relations for relational representation of the world.

The spatial or kinematic relationships between various elements are represented by the coordinate transformation matrices owned by the natural frame of each element. The hierarchical relationships between various elements are described by the hierarchical relationships between the primitives and between auxiliary frames. All these were described in the previous sections. This section discusses the representation of affixments or connectivities between different primitives making up the world model.

Affixment descriptors, within RWORLD, describe a form of ownership relations that cannot be instantiated but can be defined and modified dynamically during programming and execution. Each affixment descriptor has a attribute that describes the nature of the affixment or connectivity of two primitives within the world model. At present two types of attributes are used to represent, respectively, rigid and non-rigid affixments. The rigid affixment is used to describe, for example, grasping of parts by end-effectors or fixtures, assemblies, and device aggregations. The rigid affixment is fully supported in RWORLD.

The non-rigid affixment is used to describe the situation where one item rests on another item. The non-rigid attribute is only partially supported within RWORLD to describe situations where objects or devices are placed, in stable locations, on the top of other objects or devices. An example is one block that is placed on top of a larger block. Furthermore, the non-rigid affixment only represents the nature of the interdependence of two elements of the world from a purely kinematic viewpoint. In other words, if an object B is affixed to an object A by a non-rigid affixment, then if object A undergoes a displacement, object B also undergoes the same displacement.

The converse of this situation, however, does not hold. In this process, it is assumed that the motion of A neither change the relative position of object B with respect to object A nor does it make object B unstable relative to object A. Ideally, the non-rigid affixment should also model the nature of contact and force transmission between the contacting surfaces of the two objects.

The use of rigid affixments allows creation of assemblies by contacanating objects. It also allows creation of device aggregates at run time without the need to change or remodel the two aggregated devices as one device. This is specially important in simulating tool changes where, for example, a robot picks up a dexterous hand. In such a situation the new robot-dexterous-hand combination actually represents a new device with a totally different set of kinematic and dynamic equations.

In RWORLD, since the kinematic and dynamic equations for the robot and the dexterous hand are already developed in the world model, the rigid affixment (or aggregation) of the two is handled by considering the two devices under coordinated control. The parallel command execution technique described in the next section is then used to program the motion of the robot-dexterous-hand combination by separately using the corresponding kinematic and dynamic equations in a coordinated manner.

When using affixments, the relationships between differen components of a robot workcell can be described in the form of a tree or a graph structure. At present, RWORLD only supports the tree structure. The nodes of a tree are data elements describing one of the four primitives of devices, objects, sensors and frames. Fig. 9 shows a workcell where two dexterous hands are aggregated to a robot and the robot is aggregated to a conveyor belt, which itself has one translational degree of freedom and represents another device.

In addition, there are two objects, each grasped by one of the two dexterous hands, and with one of the objects representing an assembly of three components. There are also two auxiliary frames affixed to this assembly. A tree representation of this workcell is shown in Fig. 10.

As the rigid affixments are deleted during program run time, the tree is split into smaller trees at the corresponding nodes. This is the case, for example, when the end-effectors release the objects they are holding or if the robot performs a tool change replacing the two end-effectors by other end-of-arm toolings. The ownership relationship represented by affixment descriptors allows up-

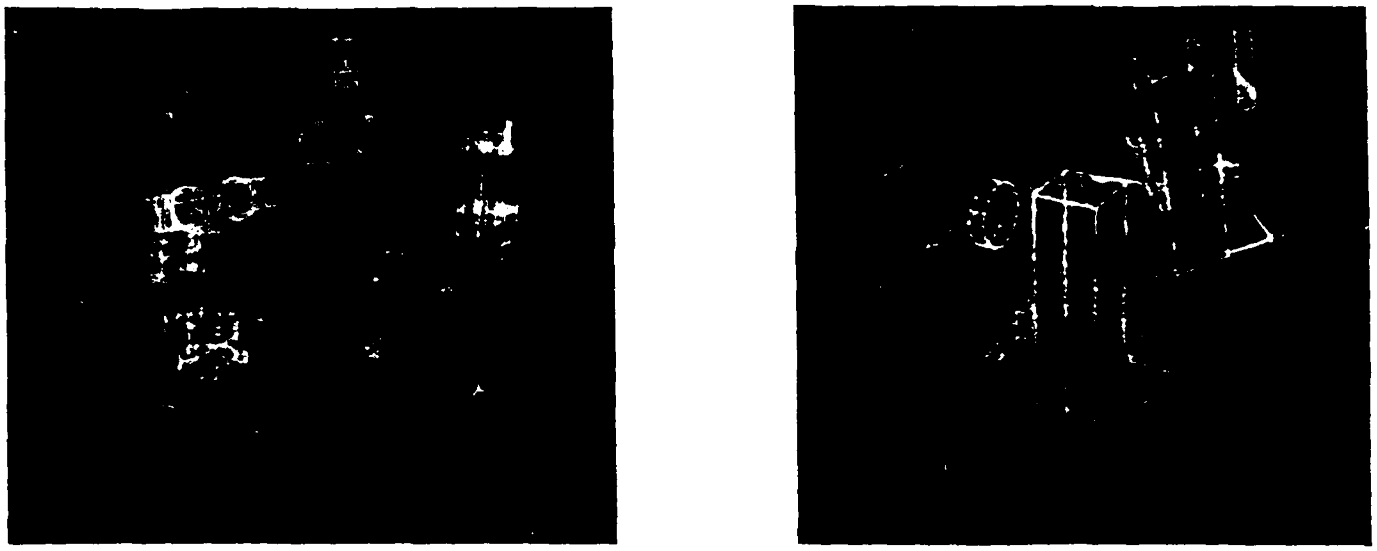


Figure 9. A robot with aggregated end-effectors before and after grasping a part and an assembly.

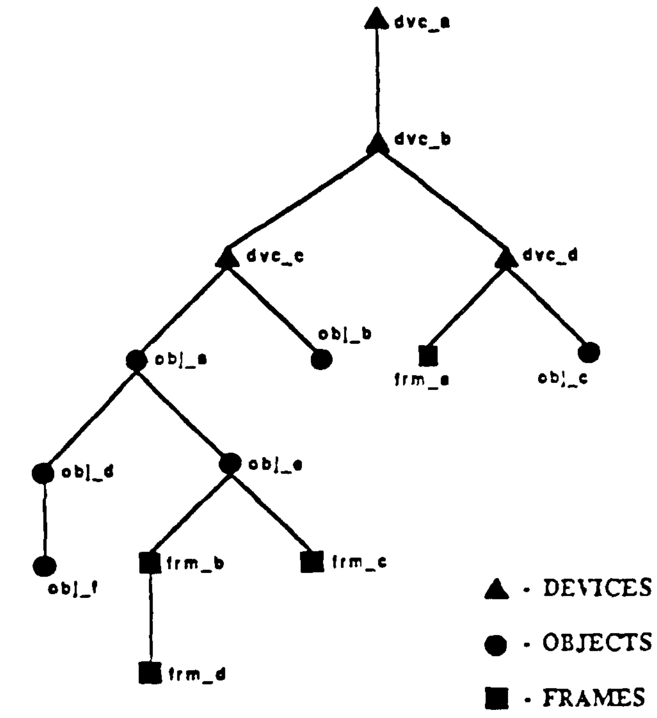


Figure 10. The tree structure of the workcell hierarchy of Figure 9 after the robot has grasped the part and the assembly.

dating the location of all nodes of the tree subordinate to the node whose location in the workcell is changed. Crude versions of some of the device aggregation ideas were also implemented in the STAR program using conventional programming languages such as "c" [8]. These implementations, however, did not use parallel operation of aggregated devices. RWORLD supports both serial and parallel execution of devices aggregated or operated cooperatively.

5. Handling of Parallelism

Parallelism, in the context of CAD-based simulation of robotic workcells, is when multiple devices are operating at the same time. The devices may be aggregated or working in parallel in a cooperative manner. In such situations the robot workcell simulation environment should be able to properly display parallel operations of these devices. This can be handled by first allowing parallel and serial aggregation of memory segments allocated to each device during execution. Then if, for example, parallel processors are used, each parallel memory aggregate can be allocated to a different processor; otherwise, the execution should be swapped between parallel aggregates. In this latter case, the time increments for the execution of the motion programs for devices that are working in parallel should be controlled.

Since certain devices are operated in series with others, and the motion programs for all parallel devices may end at different time steps, a device checker should also be used. The function of the device checker is to check the aggregation and execution status of each device. This is done during execution, either to schedule operation of a device, if it is serially operated with respect to certain other devices, or to neutralize a device from parallel operation when its task is completd. The device checker also informs the time-step controller about the devices for which the execution time should be sliced into small increments. The time-step controller then swaps the execution between these devices such that a uniform execution time between all the devices is created. This is all illustrated schematically in Fig. 11. Fig. 12 is a computer generated display of three devices (two robots and a convey belt) operated in parallel to perform a material transfer and an assembly task.

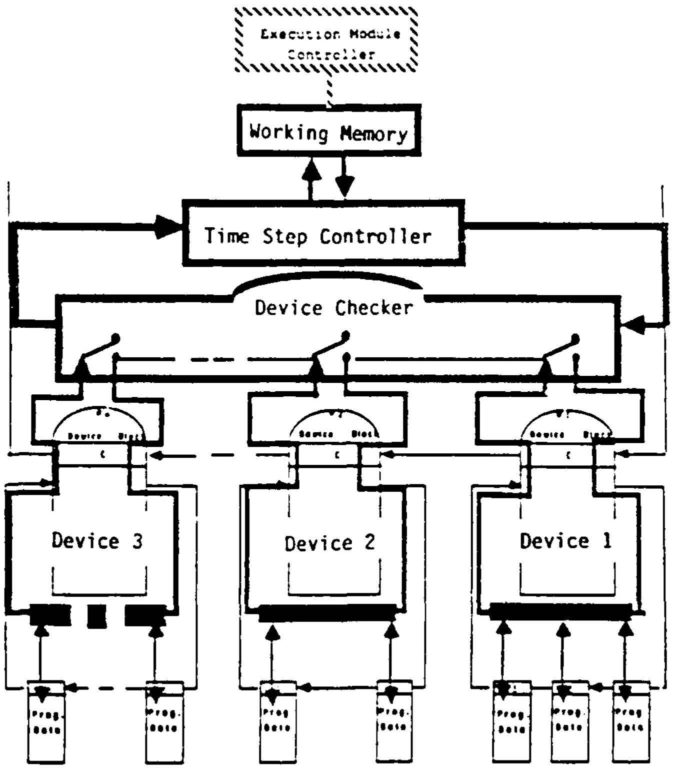


Figure 11. The execution environment for handling parallelism.

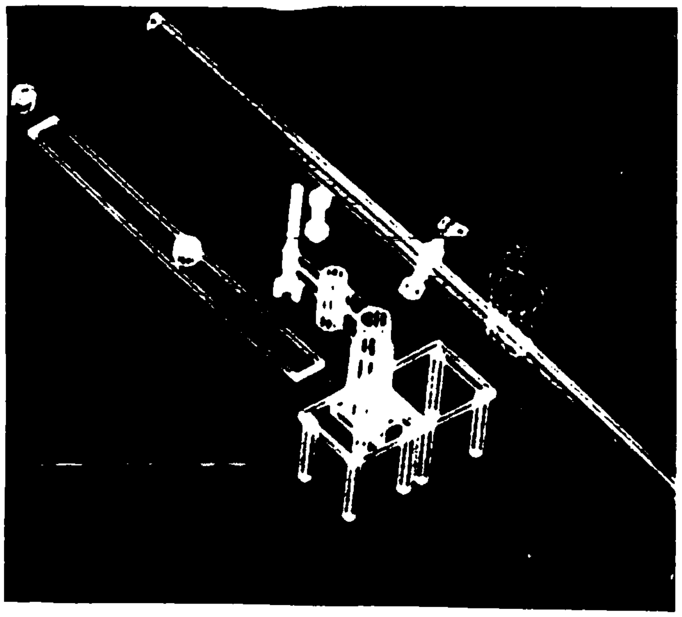


Figure 12. Computer display of parallel operation of devices.

6. World Model Calibration

In CAD-based robot programming and simulation, uncertainties are introduced due to modeling errors and tolerances on geometrical descriptions and specification of spatial relationships between objects. These errors are in addition to those introduced by sensors and actuators during task execution. The latter set of errors are control errors and can be corrected by sensory control during task execution. The first set of errors, however, are errors in the world model. These errors may be corrected by world model calibration. Calibration uses sensory information to update the world model.

Most previous studies on robot calibration [9] have been devoted to robot programming systems using simplistic world models. These systems only use the robot kinematic equations as part of the world model and, in addition, are not supported by a simulation environment or a geometric modeling system. Furthermore, regular robot calibration procedures only involve calibration of internal robot model and not all the components of the robot workcell. Calibration of all the components of the world model of a robotic workcell requires sensory interactions between the world model and the robot workcell.

Grossman and Taylor [10] have described a system where a robot itself is moved to a position within its workspace to update position or orientation data. Hasegawa [11] has developed a system that can be used to construct and calibrate position data as well as to verify geometric shape models in an interactive fashion. More recently, Ishii et al. [12] have developed a sensor that can ease up the sensing of position and orientation data in a robot workcell environment.

In RWORLD, we are developing a system that uses a combination of tactile and vision sensing for monitoring the workcell. Once sensory data is obtained from a robot workcell, it is used to modify the world model. In CAD-based programming, calibration is usually performed before the robot program is downloaded to the actual robot. This means that the already-generated robot program may also have to be modified due to the modifications of the world model. In order to make the robot program independent of the numerical data associated with the robot workcell, RWORLD uses the abstract primitive representation of the workcell in terms of devices, objects, sensors, and frames.

The numerical data on spatial relationships and physical and shape properties are all kept in separate files that can be accessed by the corresponding primitive. If the robot program is then written symbolically in terms of these primitives, update of the numerical values in the world model does not require any changes in the robot program: it only requires the update of numerical values in the corresponding numerical data files. Furthermore, since present robot-programming languages do not support many of the details associated with shape information or physical properties of the world, only calibration of spatial (or kinematic) relationships is considered in RWORLD. The system, however, is designed to support more higher level calibrations for shape or physical properties.

7. Simulation of Sensory Interactions

In CAD-based simulation of sensory robots, the world model should be able to model sensory interactions that may occur between the robot and its workcell environment. This is necessary in order to allow reasoning about error recovery in programming robots equipped with external sensors. There are two main categories of external sensing techniques, one involving local and the other using global interactions with the environment. The existing sensing functions for the first category include touch, proximity, force, and presence sensing. Vision is probably the most important external sensory function involving global (long-range) interaction with the robot environment.

In a CAD-based robot simulation system, simulation of local sensing functions involves interactions with geometric models of objects. This means that all such sensory functions can be simulated geometrically. RWORLD uses interference checking between geometric models of sensors and objects to simulate touch, tactile, presence, and proximity sensing functions. Existence of interference between the geometrical model of a sensor primitive representing a touch sensor and that of an object simulates existence of contact between the two elements. This is very simple and was suggested by Meyer [13].

Meyer also suggested the use of simple interference checking to simulate the use of a light beam between the fingers of an end-effector to check presence or absence of objects. In this case, the geometric model of the sensor should represent the geometry of the light beam. If, in addition to touch, distribution of contact (tactile information) is also to be simulated, then the problem is slightly more complicated. RWORLD partitions the geometric model of the tactile array into an array of small geometrical segments and then performs the interference checking between each segment and the object. All segments having interferences with the object are then color coded for display of distribution of contact (Fig. 13).

The simulation of proximity sensing is also very similar to this and again should use interference checking for segmented models. In this case the geometrical model of the sensor consists of a prismatic object having the same size as the volumetric range of the actual priximity sensor being simulated (Fig. 13). This geometric representation of the volumetric range of the sensor is partitioned into segments, each representing units of proximity to the object to which the actual sensor is attached. The proximity sensing function can then be simulated by checking for interferences between the segments of the sensor model and the external object. The segments having interferences with the object are then color coded for display purposes to illus'rate the remaining units of closeness to the object.

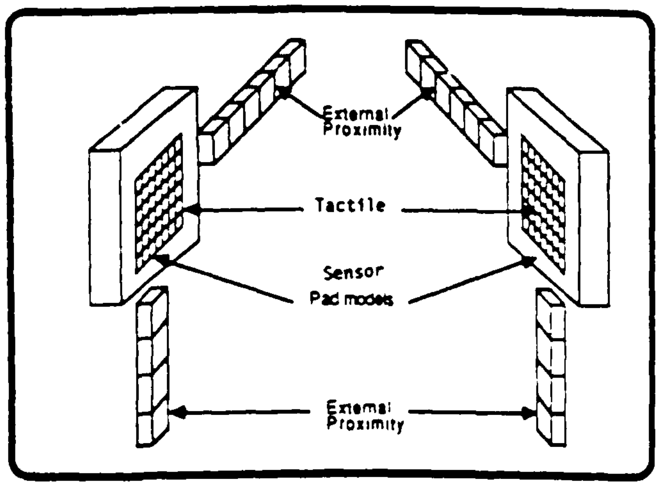


Figure 13. Geometric partitioning for simulation of tactile and proximity sensing.

Simulation of the sensory functions described so far requires algorithms for fast detection of interferences between geometric (solid) models. This is why RWORLD uses prismatic approximation of polyhedral solids in the second level of its multi-level representation of geometric shape information. Interference between prismatic solids can be computed faster than that of polyhedral solids.

A higher level of complication arises if force sensing is to be simulated geometrically. In this case volume of overlap between the geometric model of the sensor and the object should be computed in addition to checking for the existence of overlap or interference. Different amounts of overlap should also be calibrated in relation to different levels of force readings in the actual force sensor.

Simulation of global sensory interactions with the environment such as that of vision has not yet been considered for RWORLD.

8. Conclusion

This paper has discussed the basic elements of world modeling for CAD-based robot programming and simulation. The design of an advanced world modeling system, RWORLD, is also presented together with examples from partial implementations of some of the elements of the system. The overall design of RWORLD is very open ended and flexible. Its implementation can grow to a powerful but generic system that can form the basis for robotic simulation, programming, and control.

Acknowledgments

This work was supported, in part, by the US Army Research Office under grant number DAAL 03-88-k-0206. Dr. M. L. Hornick performed some of the initial implementation of the device aggregation functions using a conventional programming language.

References

[1] Y.F. Yong, J.A. Gleave, J.L. Green, and M.C. Bonney, "Off-Line Programming of Robots." In *Handbook of Industrial Robots*, ed. S.Y. Nof (New York: John Wiley, 1985): 366-380.

[2] M.L. Hornick and B. Ravani, "Computer Aided Off-Line Planning and Programming of Robot Motion." *Int'l J. of Robotics Res.*, *4* (4): 18-31, 1986.

[3] T. Lozano-Perez, "Automatic Planning of Manipulator Transfer Movement." IEEE Trans. Syst. Man. Cybern. SMC 11 (10): pp. 681-689. 1981.

[4] J. C. Latombe, " Towards Automatic Robot Programming in Control and Programming in Advanced Manufacturing ed. K. Rathmill, Int'l Trends in Manufacturing Tech. [illegible] pp. 85-102, 1988.

[5] B. Ravani, and M. L. Hornick, " STAR: A simulat[illegible]

Automation and Robotics," in Control and Programming in Advanced Manufacturing, ed. K. Rathmill, Int'l Trends in Manufacturing Tech., IFS Pub., pp. 269-294, 1988.

[6] L.T. Wang and B. Ravani, "Recursive Computations of Kinematic and Dynamic Equations for Robot Manipulators." *IEEE J. of Robotics and Automation, RA-1* (3): 124-131, 1985.

[7] L.T. Wang and B. Ravani, " Dynamic Load Carrying Capacity of Robot Manipulators - Part I: Problem Formulation." *ASME Trans., J. of Dynamic Systs., Measurem. and Control, 110:* 46-52, 1988.

[8] M.L. Hornick and B. Ravani, "A Geometric Data Base for Off-Line Robot Programming." *Proc. of the 7th World Congress on Theory of Machines and Mechanisms,* Sevilla, Spain: 1227-1231, 1987.

[9] Z.S. Roth, B.W. Mooring, and B. Ravani, "An Overview of Robot Calibration." *IEEE J. of Robotics and Automation, RA-3* (5): 377-385, 1987.

[10] D.D. Grossman and R.H. Taylor, "Interactive Generation of Object Models with a Manipulator." *IEEE Trans. Syst. Man and Cybern., SMC-8* (9): 667-679, 1978

[11] T. Hasegawa. "An Interactive System for Modeling and Monitoring a Manipulation Environment." *IEEE Trans. Syst. Man and Cybern., SMC-12* (3): 250-258, 1982.

[12] M. Ishii, S. Sakane, Y. Mikami, and M. Kakikura. "A 3-D Sensor System for Teaching Robot Paths and Environment." *Int'l J. of Robotics Res., 6* (2): 45-59, 1987.

[13] J. Meyer, "An Emulation System for Programmable Sensory Robots." *IBM J. Res. and Develop., 25* (6): 955-1981, 1981.

World Model Calibration for Off-line Robot Programming

June 20, 1989

Introduction

In CAD-based off-line programming and simulation, it is necessary to have an appropriate model of the robot world. The world model provides the necessary knowledge of the robot workcell for a task planner to generate appropriate robot motion control commands. The task planner accesses the world model when converting motion commands into trajectory directives and updates the world model when the state of the workcell configuration changes. During robot task execution, for example, parts are grasped, repositioned and released or assemblies are created. All this must be reflected within the world model.

Up to this moment, existing CAD-based off-line programming and simulation systems only work with nominal world models. Robot programs, called nominal robot programs, are created based upon these nominals models. However uncertainties exist in the physical world, uncertainties whose values cannot be exactly determined at the time the nominal world model and robot program are created. Even if one is able to measure and determine the uncertainties on a specific moment, it is possible that new uncertainties will be introduced before, or during robot program execution.

There are three major sources of uncertainties: (1) the manipulator, (2) the objects to be manipulated, and (3) the introduction of these objects into the work environment.
To eliminate or reduce the second and third source of uncertainties, the nominal world model should be calibrated before robotprogram execution. This means that we have to define the real location of a number of frames in the world model. The location of each object or device in the world model is defined by a special frame, named the natural frame of the object or device, so we can deal with the third source of uncertainties by defining the real location of these natural frames. Special characteristics of the objects (grasp locations, holes, ...) are represented by auxiliary frames. Defining the real location of these frames eliminates or reduces the second source of uncertainties.

The first source of errors can be dealt with in different ways. First, one can calibrate the kinematic model of the robot before robot program execution. Secondly, it should be possible to execute the robotprogram only once with sensory feedback (e.g. force) to determine the errors caused by the imperfect kinematic model of the manipulator. Thirdly, the robot program execution can be done with on-line sensory feedback (e.g. external force control) to deal with these errors. The last two possibilities have the extra advantage that they also reduce the errors caused by unknown uncertainties or uncertainties introduced during execution.

In this work the errors introduced by the imperfect kinematic model of the manipulator will be reduced by using the second or third method. The first method can form part of future research and can then be compared with results obtained by the two other methods.

The first part of this work will consist in developping a world model calibration procedure. This procedure will be implemented in the world modeling system RWORLD. As we will see further RWORLD uses a multi-primitive representation of the workcell environment. These primitives are devices, rigid objects, frames and sensors. To make world model calibration possible, we have to introduce a new primitive: the feature primitive. It is this feature primitive that will be measured in the physical world. The measurement of the features will be only based on point measurements.

The global procedure for creating a robot program, including calibration of the world model, can be briefly outlined as follows:
1. create a world model, consisting of devices, objects, sensors and frames
2. create a frame based robot program
3. introduce features into the world model and condition the location of important frames upon these features
4. create a world model calibration procedure
5. measure necessary features with point measurements
6. update world model and/or robot program
7. execute robot program without on-line sensory feedback
   or
7. execute robot program with on-line sensory feedback
   or
7. execute robot program once with sensory feedback, update robot program, execute robot program again without sensory feedback

Step 1 and 2 are classic steps, also occurring in existing systems. Step 3 is new and will be explained later. Step 4 can be viewed as the creation of a list with measuring directives: which features have to be measured? Step 5-7 are self-explanatory. When we say "create" in step 1, 2 and 4 we mean: create interactively with the system. However research should be done to automate the 4th step. Enough information, generated by the third step should be available to do this automation.

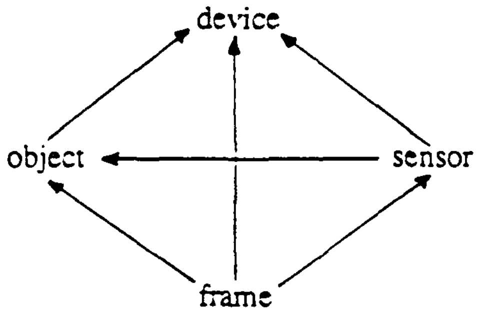
Introducing the feature primitive for world model calibration opens the door to robot programming on feature level. Today robot programming is done on frame level. Motion commands are generated based on frame locations. Research should be done to facilitate the robot programming by reasoning on feature level.

Finally a sensor device should be designed that enables us to do point measurements in space.

World model calibration

1. RWORLD

In a complex application, a manipulator may use several different tools and end-effectors, interact with equipment or manipulators, use sensory data and manipulate and process parts to create component assemblies or products. The tools or end-effectors used may be other complex manipulators such as dexterous mechanical hands. The parts may also be subassemblies of other parts. This means that robot workcells consist of hierarchical aggregations of several distinct elements such as parts, equipment and sensors each having different physical representations. A rigid part, for example does not have any internal kinematic representation while a manipulator does have an internal kinematic representation due to the relative freedom between its links. For this reason, RWORLD uses a multi-primitive representation of the workcell environment at an abstract level. These abstract primitives are devices, rigid objects, frames and sensors. The following figure illustrates the hierarchical organization of these four primitives:



A device is an entity having one or more internal degrees of freedom. Examples of devices are robots, machine tools, conveyor belts, end-effectors and flexible fixtures. A rigid object is an entity with no internal degrees of freedom but capable of being moved with six degrees of freedom in the workcell environment. Examples of objects are workpieces and tables. The two primitives of devices and objects are defined in a hierarchical fashion such that objects can be accessed by devices but devices cannot be accessed by objects. Frames are the third primitive in the world model: They are used to mark a location (position and orientation) in space. Each device, object and sensor owns a frame, referred to as natural frame, that represent its location in space. In addition to their use as natural frames, frame primitives are also used to mark characteristic locations on objects or devices. Such frames are referred to as auxiliary frames. Sensors are the last primitive within the world model. A sensor primitive provides a functional representation for sensory interactions for simulation purposes.

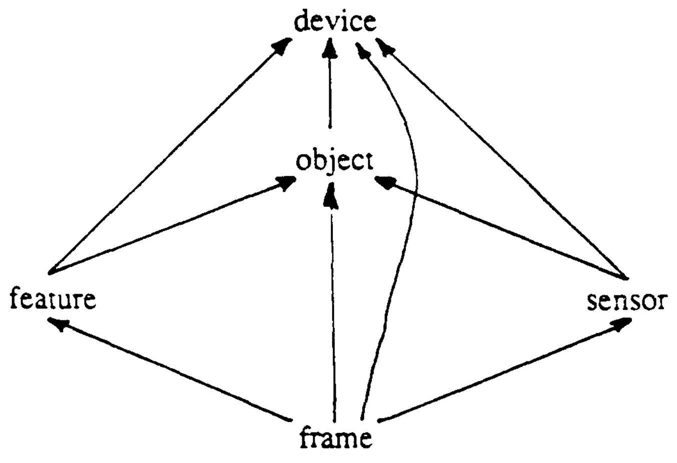
2. The feature primitive

The calibration of the nominal world model consists in defining the real locations of important natural and auxiliary frames in the world model. Therefore measurements have to be carried out on the physical robot workcell. As stated in the introduction, at this moment we only allow point measurements. Future research may investigate other possible measurement methods. Whatever method of measurement is used, it will always be subject to errors. Therefore a least squares approximation approach will be utilized to minimize these errors.

The location of the natural frame of an object in 3D-space is totally defined by the position of three non-collinear points on the object, so we would have enough information to find the real location by measuring three points on the object. To eliminate or reduce the errors introduced by the measurement of the three points, we measure more than three points and do a least squares approximation. The next question that arises, is which multiple points to measure. Because we are dealing here with CAD-based off-line programming, we can assume that we have a geometric model of the object. So we can assume that we know the equations of several features (surfaces, curves, ...) of the object relative to the object's natural frame. Therefore we will measure multiple points on features and we will define the equations of these features with a least squares approximation approach to reduce the measurement errors.

So to calibrate a nominal world model, we will only perform point measurements on features. These point measurements will enable us to define the real equations of the features relative to the world frame and the real equations of these features will enable us to find the real locations of natural and also of auxiliary frames. It is this information, the real locations of a number of frames, that we are looking for in world model calibration, because up to this moment robot programming is based on frame information, not on feature information.

To define the real locations of these important frames by point measurements on features we have to connect these frames to features. A frame will be connected to a feature by a condition. The feature will as such condition the frame (e.g. the (X,Y)-plane of the frame has to be parallel to a plane feature). Therefore, a feature primitive, which is a geometric characteristic of an object or a device and is owned by that object or device, will be introduced into the hierarchical organization of RWORLD primitives:



A frame can be connected to a feature which imposes a condition on the frame. In that case the frame and the feature have to belong to the same object or device.

3. Defining the equations of features by point measurements

To define the equations of features we will perform point measurements on these features. To eliminate or reduce the measurement errors, we will do a least squares approximation. In this chapter we will discuss the plane feature.

In many physical and statistical investigations it is desirable to represent a system of points in space by the "best-fitting" plane. Analytically this consists in taking

$$z = a_0 + a_1 \cdot x + a_2 \cdot y$$

where x, y and z are variables, and determining the "best" values for the constants a_0, a_1 and a_2 in relation to the observed corresponding values of the variables. In nearly all cases dealt with in text-books of least squares, the variables on the right of our equations are treated as the independent, those on the left as the dependent variables. In many cases however, as in our case, the "independent" variable is subject to just as much deviation or error as the "dependent" variable. In these cases, we will consider as the "best fit", the plane that minimizes the sum of the squares of the perpendiculars from the system of points upon the plane.

So let P_1, P_2, ..., P_n be the system of points with coordinates (x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n) and perpendicular distances p_1, p_2, ..., p_n from the plane. Then we shall make

$$U = \Sigma(p^2) = \text{a minimum.}$$

Let $\bar{x} = \Sigma(x_i)/n$, $\bar{y} = \Sigma(x_i)/n$ and $\bar{z} = \Sigma(x_i)/n$ (1)

be the mean values of the variables;

$$\sigma_x^2 = \Sigma(x_i^2)/n - \bar{x}^2,\ \sigma_y^2 = \Sigma(y_i^2)/n - \bar{y}^2 \text{ and } \sigma_z^2 = \Sigma(z_i^2)/n - \bar{z}^2 \tag{2}$$

be the standard-deviations, and, lastly, let

$$r_{xy} = \frac{\Sigma(x_i y_i) - n\bar{x}\bar{y}}{n\sigma_x\sigma_y},\ r_{xz} = \frac{\Sigma(x_i z_i) - n\bar{x}\bar{z}}{n\sigma_x\sigma_z} \text{ and } r_{yz} = \frac{\Sigma(y_i z_i) - n\bar{y}\bar{z}}{n\sigma_y\sigma_z}, \tag{3}$$

be the correlations of the variables.

Now let l_1, l_2 and l_3 be the direction cosines of a plane

$$l_1x + l_2y + l_3z = p \tag{4}$$

at a perpendicular distance p from the origin. We shall have

$$l_1^2 + l_2^2 + l_3^2 = 1 \tag{5}$$

Then we require to make a minimum of

$$U = \Sigma(l_1x + l_2y + l_3z - p)^2 \tag{6}$$

by variation of l_1, l_2, l_3 and p subject to (5). The Lagrangian multiplier theorem is used to minimize U under the constraint (5). To use this theorem a Lagrangian multiplier Q and a function U^* are introduced:

$$U^* = \Sigma(l_1x + l_2y + l_3z - p)^2 - Q(l_1^2 + l_2^2 + l_3^2 - 1)$$

Differentiation with regard to p gives:

$$l_1\Sigma(x_i) + l_2\Sigma(y_i) + l_3\Sigma(y_i) - np = 0$$

or

$$p = l_1\bar{x} + l_2\bar{y} + l_3\bar{z} \tag{7}$$

which shows us that the best fitting plane passes through the centroid of the system.

Differentiation with regard to l_1, l_2 and l_3 gives:

$$\frac{\partial U}{\partial l_1} = 0 = l_1\Sigma(x_i^2) + l_2\Sigma(x_iy_i) + l_3\Sigma(x_iz_i) - p\Sigma(x_i) - Ql_1$$

$$\frac{\partial U}{\partial l_2} = 0 = \ldots$$

$$\frac{\partial U}{\partial l_3} = 0 = \ldots$$

substituting for p from (7) gives:

$$l_1\Sigma(x_i^2) + l_2\Sigma(x_iy_i) + l_3\Sigma(x_iz_i) - l_1\bar{x}\Sigma(x_i) - l_2\bar{y}\Sigma(x_i) - l_3\bar{z}\Sigma(x_i) - Ql_1 = 0$$

using (2) and (3) gives:

$$\begin{aligned} l_1\sigma_x^2 + l_2\sigma_x\sigma_y r_{xy} + l_3\sigma_x\sigma_z r_{xz} - (Q/n)l_1 &= 0 \\ l_1\sigma_x\sigma_y r_{xy} + l_2\sigma_y^2 + l_3\sigma_y\sigma_z r_{yz} - (Q/n)l_2 &= 0 \qquad (8) \\ l_1\sigma_x\sigma_z r_{xz} + l_2\sigma_y\sigma_z r_{yz} + l_3\sigma_z^2 - (Q/n)l_3 &= 0 \end{aligned}$$

These are the type equations.

We can proof that $Q = U_m$ = minimum value of U. Let S be the mean square of the residuals:

$$S = \frac{\Sigma(l_1x + l_2y + l_3z - p)^2}{n}$$

Then: $Q/n = S$; this gives a physical meaning to Q.

The determinantal equations of the type equations:

$$\begin{vmatrix} (\sigma_x^2 - S) & \sigma_x\sigma_y r_{xy} & \sigma_x\sigma_z r_{xz} \\ \sigma_x\sigma_y r_{xy} & (\sigma_y^2 - S) & \sigma_y\sigma_z r_{yz} \\ \sigma_x\sigma_z r_{xz} & \sigma_y\sigma_z r_{yz} & (\sigma_z^2 - S) \end{vmatrix} = 0$$

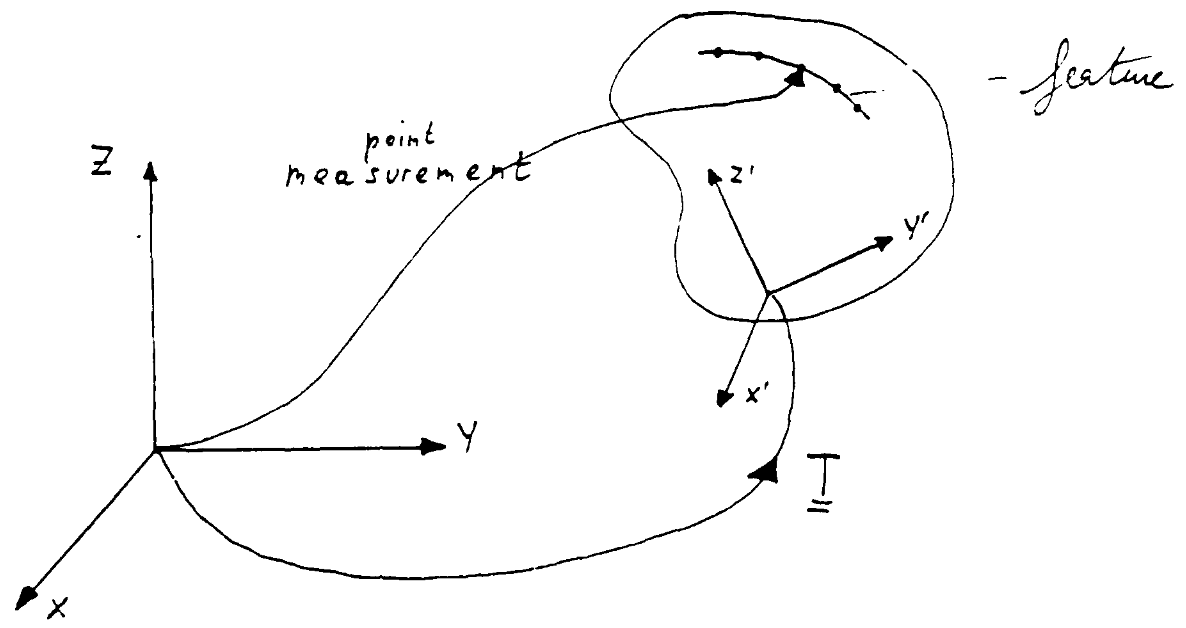
or

$$\begin{vmatrix} (1 - S/\sigma_x^2) & r_{xy} & r_{xz} \\ r_{xy} & (1 - S/\sigma_x^2) & r_{yz} \\ r_{xz} & r_{yz} & (1 - S/\sigma_x^2) \end{vmatrix} = 0$$

We must choose the least root of this equation, for the mean square residual must be as small as possible. Substitute this value of S in the type equations and we find the required values of l_1, l_2 and l_3 using (5).

A least squares approximation of a straight line can be done in a similar manner.

4. Natural frames



Defining the real location of the natural frame of an object is equivalent to defining the transformation matrix $\underline{T}$ of that frame relative to the world frame. As stated above, we will perform point measurements on features. Up to this moment, in defining the real location of natural frames, we will only use planes and straight lines as features. The measurements will be performed relative to the world frame, so they will give us the equations of the planes and lines relative to the world frame. From the geometrical model of the object we also have to our disposal the equations relative to the object's natural frame.
These equations, relative to the world frame and relative to the object's natural frame, form the input to an algorithm that will calculate the transformation matrix.

Measuring points on a line or plane in an arbitrary way, will result in equations without orientation information. If we need orientation information, we have to measure the features in a specific way. For a line, the first and last point will define the orientation of the line; for a plane, the last point will be a point above the plane (above means outside the object).

So we have the following equations:

- straight line: $x = x_p + r \cdot a$
  $y = y_p + r \cdot b$
  $z = z_p + r \cdot c$

  or: (x_p, y_p, z_p) and $\text{sign} \cdot (a, b, c)$

- plane: $a \cdot x + b \cdot y + c \cdot z + d = 0$

  or: $\text{sign} \cdot (a, b, c, d)$

With sign = ± 1 and sign will only be known if we have orientation information.

In the following, we make distinction between the case with orientation information and the case without.

4.1. Features with orientation information

4.1.1. Necessary and sufficient information to define $\underline{T}$

In this paragraph we define the minimum number of features we have to measure.

4.1.1.1. General

- in the following $\underline{a}$ is a vector relative to the world frame and $\underline{a}'$ a vector relative to the object's natural frame
- we divide the problem in finding the rotation matrix $\underline{R}$ and the translation vector $\underline{t}$ with: $\underline{T} = \begin{bmatrix} \underline{R} & \underline{t} \\ 0 & 1 \end{bmatrix}$

- for $\underline{R}$ we need 2 direction vectors

  proof: - given $\underline{a}_1 = (a_{1x}, a_{1y}, a_{1z})^T$ and $\underline{a}_1'$
  $\underline{a}_2 = (a_{2x}, a_{2y}, a_{2z})^T$ and $\underline{a}_2'$
  - calculate: $\underline{a}_3 = \underline{a}_1 \times \underline{a}_2$
  $\underline{a}_3' = \underline{a}_1' \times \underline{a}_2'$
  - then: $\begin{bmatrix} a_{1x} & a_{2x} & a_{3x} \\ a_{1y} & a_{2y} & a_{3y} \\ a_{1z} & a_{2z} & a_{3z} \end{bmatrix} = \underline{R} \cdot \begin{bmatrix} a_{1x}' & a_{2x}' & a_{3x}' \\ a_{1y}' & a_{2y}' & a_{3y}' \\ a_{1z}' & a_{2z}' & a_{3z}' \end{bmatrix}$
  - or: $\underline{A} = \underline{R} \cdot \underline{A}'$
  - or: $\underline{R} = \underline{A} \cdot \underline{A}'^{-1}$

- for $\underline{t}$: we need 1 position vector

  proof: - given $\underline{p} = (p_x, p_y, p_z)$ and $\underline{p}'$
  - then: $\underline{p} = \underline{R} \cdot \underline{p}' + \underline{t}$
  - or: $\underline{t} = \underline{p} - \underline{R} \cdot \underline{p}'$

4.1.1.2. Points

- if we only measure points, we need minimum 3 non-collinear points

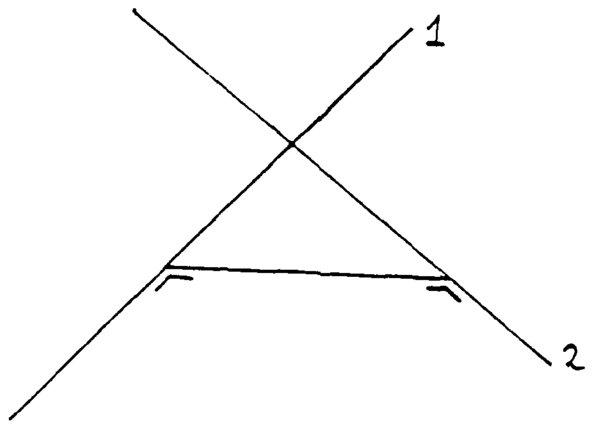
- proof: - given: $\underline{a}$, $\underline{b}$, $\underline{c}$ and $\underline{a}'$, $\underline{b}'$, $\underline{c}'$
  - we can construct two direction vectors: $\underline{r}_1 = \underline{b} - \underline{a}$
    $\underline{r}_2 = \underline{c} - \underline{a}$

  so we can find $\underline{R}$
  - since we have 2 position vectors, we can find $\underline{t}$

4.1.1.3. Straight lines

- if we only measure straight lines, we need minimum 2 non-parallel straight lines

- proof: - given: - line 1: direction vector $\underline{a}_1$ and a point $\underline{p}_1$ on the line (and also $\underline{a}_1'$ and a different point $\underline{p}'$)
  - line 2: $\underline{a}_2$ and $\underline{p}_2$
  - we have 2 direction vectors, so we can find $\underline{R}$
  - we can calculate 1 specific point on one of the lines : e.g. the point on line 1 closest to line 2 (see fig.); so we can find $\underline{t}$



4.1.1.4. Planes

- if we only measure planes, we need minimum 3 non-parallel planes

- proof: - the intersection of the 3 non-planar planes results in 3 non-parallel straight lines, so we have enough information (see 4.1.1.3.)

4.1.1.5. Combination of straight lines and planes

a) 1 Straight line and 1 plane
- sufficient if the straight line is not parallel nor perpendicular to the plane

- proof: - $\underline{R}$: take direction vector of the straight line and the perpendicular to the plane
  - $\underline{t}$: calculate intersecting point of the straight line with the plane
- because the 2 conditions on the straight line and the plane are somewhat contradictory, 1 straight line and 1 plane are sufficient, but the algorithm will be numerically instable.

b) 1 Line and 2 planes
- sufficient if the straight line is non-parallel to the intersection of the planes
- proof: - with the straight line and the intersection of the planes we have two straight lines (see 4.1.1.3.)

4.1.2. Defining $\underline{T}$ with 3 or more points

See 4.2.2.

4.1.3. Defining $\underline{T}$ with 3 or more planes

- given: - for i = 1, n
  - $(q_{xi}, q_{yi}, q_{zi}, q_{ci})$ from plane $q_{xi}\cdot x + q_{yi}\cdot y + q_{zi}\cdot z + q_{ci} = 0$ measured relative to the world frame
  - $(a_{xi}, a_{yi}, a_{zi}, a_{ci})$ from plane $a_{xi}\cdot x + a_{yi}\cdot y + a_{zi}\cdot z + a_{ci} = 0$ measured relative to the object frame

- then:
$$\begin{bmatrix} q_{x1} & q_{y1} & q_{z1} & q_{c1} \\ q_{x2} & q_{y2} & q_{z2} & q_{c2} \\ \dots \\ q_{xn} & q_{yn} & q_{zn} & q_{cn} \end{bmatrix} = \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} & a_{c1} \\ a_{x2} & a_{y2} & a_{z2} & a_{c2} \\ \dots \\ a_{xn} & a_{yn} & a_{zn} & a_{cn} \end{bmatrix} \cdot \underline{T}^{-1} \qquad (9)$$

(for proof see Appendix 1)

or: $\underline{Q}'^{T} = \underline{A}'^{T} \cdot \underline{T}^{-1}$

- $\underline{T} = \begin{bmatrix} \underline{R} & \underline{t} \\ 0 & 1 \end{bmatrix}$ with $\underline{R}$ orthogonal, so $\underline{R}^{T} \cdot \underline{R} = \underline{I}$

so $\underline{T}^{-1} = \begin{bmatrix} \underline{R}^{T} & -\underline{R}^{T} \cdot \underline{t} \\ 0 & 1 \end{bmatrix}$

- substituting into (9) gives:

$$\begin{bmatrix} q_{x1} & q_{y1} & q_{z1} \\ q_{x2} & q_{y2} & q_{z2} \\ \dots \\ q_{xn} & q_{yn} & q_{zn} \end{bmatrix} = \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} \\ a_{x2} & a_{y2} & a_{z2} \\ \dots \\ a_{xn} & a_{yn} & a_{zn} \end{bmatrix} \cdot \underline{R}^{-1}$$

and

$$\begin{bmatrix} q_{c1} \\ q_{c2} \\ \dots \\ q_{cn} \end{bmatrix} = - \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} \\ a_{x2} & a_{y2} & a_{z2} \\ \dots & & \\ a_{xn} & a_{yn} & a_{zn} \end{bmatrix} \cdot \underline{R}^T \cdot \underline{t} + \begin{bmatrix} a_{c1} \\ a_{c2} \\ \dots \\ a_{cn} \end{bmatrix}$$

- this gives two sets of equations:

$$\begin{bmatrix} q_{x1} & q_{x2} & \dots & q_{xn} \\ q_{y1} & q_{y2} & & q_{yn} \\ q_{z1} & q_{z1} & \dots & q_{zn} \end{bmatrix} = \underline{R} \cdot \begin{bmatrix} a_{x1} & a_{x2} & \dots & a_{xn} \\ a_{y1} & a_{y2} & & a_{yn} \\ a_{z1} & a_{z2} & \dots & a_{zn} \end{bmatrix} \qquad (10)$$

and

$$\begin{bmatrix} q_{c1} \\ q_{c2} \\ \dots \\ q_{cn} \end{bmatrix} = - \begin{bmatrix} q_{x1} & q_{y1} & q_{z1} \\ q_{x2} & q_{y2} & q_{z2} \\ \dots & & \\ q_{xn} & q_{yn} & q_{zn} \end{bmatrix} \cdot \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} a_{c1} \\ a_{c2} \\ \dots \\ a_{cn} \end{bmatrix} \qquad (11)$$

we will solve the two sets of equations separately

4.1.3.1. Solving for $\underline{R}$

(10) can be written in the following way:

$$\begin{bmatrix} \underline{q}_1 & \underline{q}_2 & \dots & \underline{q}_n \end{bmatrix} = \underline{R} \cdot \begin{bmatrix} \underline{a}_1 & \underline{a}_2 & \dots & \underline{a}_n \end{bmatrix}$$

or $\quad \underline{q}_i = \underline{R} \cdot \underline{a}_i$ for $i = 1, n$

or $\quad \underline{Q} = \underline{R} \cdot \underline{A}$

We have to find the rotation matrix $\underline{R}$ such that $\underline{R}^T \cdot \underline{R} = \underline{I}$

Applying least squares approximation we can define the following function that has to be minimized:

$$f(\underline{R}) = \Sigma(\underline{R} \cdot \underline{a}_i - \underline{q}_i)^T (\underline{R} \cdot \underline{a}_i - \underline{q}_i)$$

The Lagrangian multiplier theorem is used to determine the matrix $\underline{R}$ that minimizes f under the constraint condition $\underline{R}^T \cdot \underline{R} = \underline{I}$. To use this theorem a (3x1) vector $\underline{S}$ of Lagrangian multipliers and a function F of $\underline{R}$ and $\underline{S}$ are introduced:

$$F(\underline{R}, \underline{S}) = f(\underline{R}) + \underline{S}^T \cdot (\underline{R}^T \cdot \underline{R} - \underline{I}) \cdot \underline{S}$$

To find $\underline{R}$ and $\underline{S}$ we take the partial derivatives of F with respect to $\underline{R}$ and $\underline{S}$ and equal them to zero.

$$\frac{\partial \underline{F}}{\partial \underline{S}} = 0 \text{ gives } \underline{R}^T \cdot \underline{R} = \underline{I}$$

$$\frac{\partial \underline{F}}{\partial \underline{S}} = 0 \text{ gives:}$$

$$F = \Sigma(\underline{a}_i^T \cdot \underline{R}^T \cdot \underline{R} \cdot \underline{a}_i) - \Sigma(\underline{a}_i^T \cdot \underline{R}^T \cdot \underline{q}_i) - \Sigma(\underline{q}_i^T \cdot \underline{R} \cdot \underline{a}_i) + \Sigma(\underline{q}_i^T \cdot \underline{q}_i) + \underline{S}^T \cdot (\underline{R}^T \cdot \underline{R} - \underline{I}) \cdot \underline{S}$$

$$\frac{\partial \underline{F}}{\partial \underline{R}} = 0 = 2 \cdot \Sigma(\underline{R} \cdot \underline{a}_i \cdot \underline{a}_i^T) - \Sigma(\underline{q}_i \cdot \underline{a}_i^T) - \Sigma(\underline{q}_i \cdot \underline{a}_i^T) + 0 + 2 \cdot (\underline{R} \cdot \underline{S} \cdot \underline{S}^T)$$

$$2 \cdot \underline{R} \cdot \Sigma(\underline{a}_i \cdot \underline{a}_i^T) - 2 \cdot \Sigma(\underline{q}_i \cdot \underline{a}_i^T) + 2 \cdot \underline{R} \cdot \underline{S} \cdot \underline{S}^T = 0$$

$$\underline{R} \cdot \underline{A} \cdot \underline{A}^T - \underline{Q} \cdot \underline{A}^T + \underline{R} \cdot \underline{S} \cdot \underline{S}^T = 0$$

$$\underline{R} \cdot (\underline{A} \cdot \underline{A}^T + \underline{S} \cdot \underline{S}^T) = \underline{Q} \cdot \underline{A}^T \qquad (12)$$

with $\underline{S}' = \underline{A} \cdot \underline{A}^T + \underline{S} \cdot \underline{S}^T$

$\underline{M} = \underline{Q} \cdot \underline{A}^T$

(12) becomes: $\underline{M} = \underline{R} \cdot \underline{S}'$

with $\underline{R}^T \cdot \underline{R} = \underline{I}$ it follows that:

$$\underline{S}'^2 = \underline{S}'^T \cdot \underline{R}^T \cdot \underline{R} \cdot \underline{S}' = \underline{M}^T \cdot \underline{M}$$

$\underline{M}^T \cdot \underline{M}$ is a symmetric matrix with eigenvalues $D_{11}^2 \geq D_{22}^2 \geq D_{33}^2 \geq 0$ and a corresponding set of three orthogonal eigenvectors. The eigenvalues are arranged on the principal diagonal of an matrix $\underline{D}^2$ while the eigenvectors are considered as the columns of a (3x3) matrix $\underline{V}$. From the definition of eigenvalues and eigenvectors it is seen that:

$$\underline{M}^T \cdot \underline{M} = \underline{S}'^2 = \underline{V} \cdot \underline{D}^2 \cdot \underline{V}^T \text{ and } \underline{V} \cdot \underline{V}^T = \underline{I}$$

A solution for the symmetric matrix $\underline{S}'$ is therefore given by $\underline{S}' = \underline{V} \cdot \underline{D} \cdot \underline{V}^T$, where the signs of the principal diagonal components D_{11}, D_{22} and D_{33} of $\underline{D}$ are positive (this can be proofed).

Insertion of this solution into $\underline{M} = \underline{R} \cdot \underline{S}'$ gives:

$$\underline{M} = \underline{R} \cdot \underline{V} \cdot \underline{D} \cdot \underline{V}^T$$

$$\text{or: } \underline{R} \cdot \underline{V} \cdot \underline{D} = \underline{M} \cdot \underline{V} = \begin{bmatrix} \underline{m}_1 & \underline{m}_2 & \underline{m}_3 \end{bmatrix}$$

As $\underline{D}$ is a diagonal matrix, column 1 and column 2 of $\underline{R} \cdot \underline{V}$ are equal to $(1/D_{11}) \cdot \underline{m}_1$ and $(1/D_{22}) \cdot \underline{m}_2$ respectively. The third column of $\underline{R} \cdot \underline{V}$ follows from the observation that $\underline{R} \cdot \underline{V}$ is orthogonal. So the final result for $\underline{R}$ is given by:

$$\underline{R} = \left[(1/D_{11})\cdot\underline{m}_1 \quad (1/D_{22})\cdot\underline{m}_2 \quad (1/D_{11}\cdot D_{22})\cdot(\underline{m}_1 x \underline{m}_2) \right] \cdot\underline{V}^T \tag{13}$$

Concluding, the procedure for the calibration of $\underline{R}$ is the following:

1) determine $\underline{M} = \underline{Q}\cdot\underline{A}^T$

2) determine the eigenvalues and eigenvectors of $\underline{M}^T\cdot\underline{M}$

3) calculate $\underline{m}_1$ $\underline{m}_2$ $\underline{m}_3$ of $\underline{M}\cdot\underline{V}$

4) calculate $\underline{R}$ according to (13).

4.1.3.2. Solving for $\underline{t}$

(11) can be written as:

$$\begin{bmatrix} q_{x1} & q_{y1} & q_{z1} \\ q_{x2} & q_{y2} & q_{z2} \\ \dots & & \\ q_{xn} & q_{yn} & q_{zn} \end{bmatrix} \cdot \begin{bmatrix} t_y \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} a_{c1} - q_{c1} \\ a_{c2} - q_{c2} \\ \dots \\ a_{c1} - q_{c1} \end{bmatrix}$$

or: $$\underline{Q}^T\cdot\underline{t} = \underline{d}$$

This is an overdetermined set of equations: we can solve this with a least squares approximation. The normal equations are:

$$(\underline{Q}\cdot\underline{Q}^T)\cdot\underline{t} = (\underline{Q}\cdot\underline{d})$$

$$\text{so } \underline{t} = (\underline{Q}\cdot\underline{Q}^T)^{-1}\cdot(\underline{Q}\cdot\underline{d})$$

The solution of a least squares problem directly from the normal equations is rather susceptible to roundoff error. An alternative, and preferred, technique involves Singular Value Decomposition, the benefits of which we may investigate in the future.

4.1.3.3. Remark

Another possible way of calculating the transformation matrix is not to take in consideration the orthogonality constraint; then we don't have to split the calculation up into two steps, we can calculate $\underline{T}$ in one step with a least squares approximation.

Starting from equation (9) we have:

$$\underline{Q}'^T = \underline{A}'^T\cdot\underline{T}'$$

$$\underline{Q}'^{T} \cdot \underline{T} = \underline{A}'^{T}$$

$$\underline{T}^{T} \cdot \underline{Q}' = \underline{A}'$$

$$\underline{T}^{T} = \underline{A}' \cdot \underline{Q}'^{T} \cdot (\underline{Q}' \cdot \underline{Q}'^{T})^{-1}$$

$$\underline{T} = (\underline{Q}' \cdot \underline{Q}'^{T})^{-1} \cdot \underline{Q}' \cdot \underline{A}'^{T}$$

The same remark can be made here, that it is better to use Singular Value Decomposition.

The advantage of this technique is that the algorithm is shorter and that it can be used for 4.1.2. with no changes. The disadvantage is that we don't have any assurance on the orthogonality of $\underline{R}$. Therefore research should be performed to investigate what will happen kinematically if we represent the location of frames by a transformation matrix with a nearly orthogonal matrix and both methods (method with and without orthogonality assurance) should be compared by an accuracy and sensitivity analysis. We can do this in the following way:

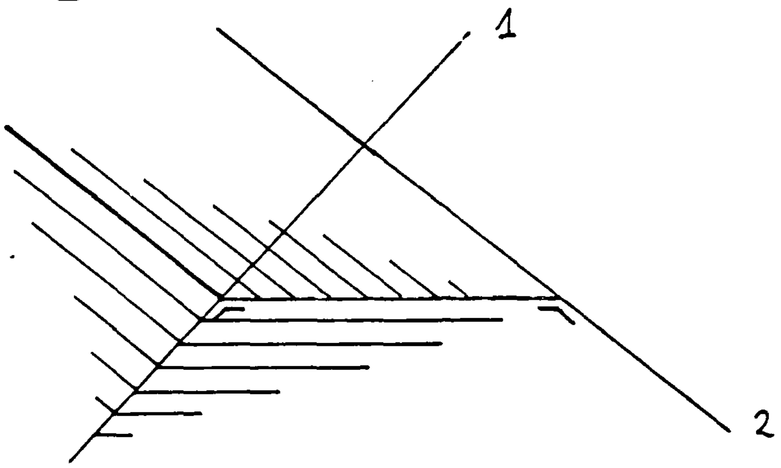
Accuracy analysis

- generate with a random number generator (RNG) a translation vector and 3 Euler angles out of which we can calculate an orthogonal matrix $\underline{R}$; this gives us a transformation matrix $\underline{T}$
- generate with the RNG the equations of 5 planes; this provides us with $\underline{Q}'$
- transform $\underline{Q}'$ into $\underline{A}'$
- with $\underline{A}'$ and $\underline{Q}'$, use both methods to define $\underline{T}_a$
- compare: $\dfrac{(t_{aij} - t_{ij})}{|tij|} \cdot 10^{-6}$
- check orthogonality: calculate $\underline{R}_a^{T} \cdot \underline{R}_a$

Sensitivity analysis

- calculate $\underline{T}$, $\underline{A}'$ and $\underline{Q}'$ as in the accuracy analysis
- introduce a relative error ε in $\underline{Q}'$ with $0.1\% \leq \varepsilon < 1.0\%$, so $\underline{Q}'_\varepsilon = (1+\varepsilon) \cdot \underline{Q}'$ (ε will be generated by a RNG)
- with $\underline{A}'$ and $\underline{Q}'_\varepsilon$, use both methods to define $\underline{T}_\varepsilon$
- compare: $\dfrac{(t_{\varepsilon ij} - t_{ij})}{|t_{ij}|} \cdot 10^{-6}$
- check orthogonality: calculate $\underline{R}_\varepsilon^{T} \cdot \underline{R}_\varepsilon$

4.1.4. Defining T with 2 straight lines



- construct 3 planes: • containing line 1 and perpendicular to line 2
  • containing line 2 and perpendicular to line 1
  • containing line 1 and parallel to line 2
- remark: the two lines may be intersecting

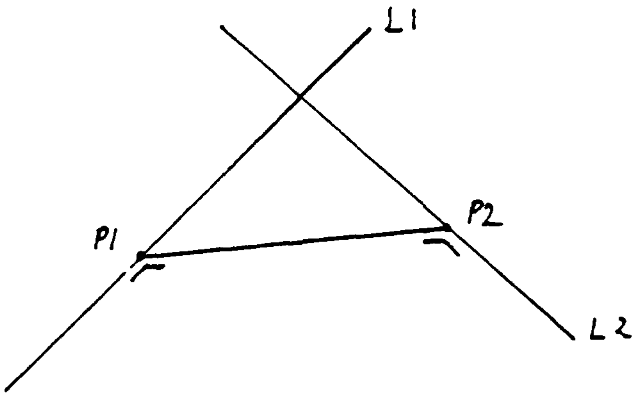
4.1.5. Defining T with more than 2 lines or with a combination of lines and planes

Reduce all the information to planes as in 4.1.4.
Define T following 4.1.3.

4.2. Features without orientation information

4.2.1. Necessary and sufficient information

If we only have features without orientation information, we will compute intersection points between the different features and we will define T with this intersection points. Following 4.1.1.2 we need a minimum number of 3 non-collinear points to define T.
- 4 non-parallel planes will yield 3 non-collinear intersection points
- under the following conditions (see fig.) 3 straight lines will yield at least 3 non-collinear points



- 2 of the lines have to be scew
- the third line may be any line except the line through P1 parallel to L2 and the line through P2 parallel to L1

4.2.2. Defining $\underline{T}$ with 3 or more points

First we have to compute at least 3 intersection points. These means that we need algorithms to compute the following:
- the intersection point between two intersecting lines
- the intersection point between a line and a plane
- the intersecting line between two planes
- the two closest points between two scew lines

The algorithm to define $\underline{T}$ with 3 or more points is similar to that one with three or more planes (see 4.1.3.). It is based on the same concepts, so we will not explain the algorithm in detail here.

5. Auxiliary frames and condition types

The previous paragraph dealt with the first part of the world model calibration: defining the real locations of the natural frames in the world model. This first part eliminates or reduces the errors caused by the introduction of the objects into the work environment.

The second part of the world model calibration deals with the objects themselves. It consists in defining the real locations of auxiliary frames. These auxiliary frames represent special characteristics of the objects. In defining the real locations of auxiliary frames, we will also use circle features.

As mentioned in chapter 2, to define the real locations of the frames, we will connect the frames to features by imposing a condition on the frame.

A natural frame will be connected to several features and can as such be conditioned by 2 condition types:

C1: determines the transformation matrix of the frame without considering orientation

C2: determines the transformation matrix of the frame with considering orientation

An auxiliary frame can only be connected to 1 feature and as such be conditioned by the following condion types:

C3: imposes the origin of the frame to lie on the line or plane feature

C4: imposes the X-axis of the frame to be perpendicular to the plane feature

C5: imposes the Y-axis of the frame to be perpendicular to the plane feature

C6: imposes the Z-axis of the frame to be perpendicular to the plane feature

C7: imposes the (X,Y)-plane of the frame to be perpendicular to the line or plane feature

C8: imposes the (X,Z)-plane of the frame to be perpendicular to the line or plane feature

C9: imposes the (Y,Z)-plane of the frame to be perpendicular to the line or plane feature

C10: imposes the (X,Y)-plane of the frame to be parallel to the plane feature

C11: imposes the (X,Z)-plane of the frame to be parallel to the plane feature

C12: imposes the (Y,Z)-plane of the frame to be parallel to the plane feature

C13: pointing to a circle feature, imposes the following conditions on a frame:
- origin of the frame must be identical to the center of the circle
- (X,Y)-plane of the frame must be identical to the plane of the circle

You can find the algorithm for C1 and C2 in chapter 4.

For each other condition (C3 - C13), used on auxiliary features, we have to develop an algorithm that will adjust the transformation matrix of the frame to the condition imposed by the feature.
For instance, with C3, the origin of the frame will be projected perpendiculary on the feature and the frame will be translated accordingly.

6. Example

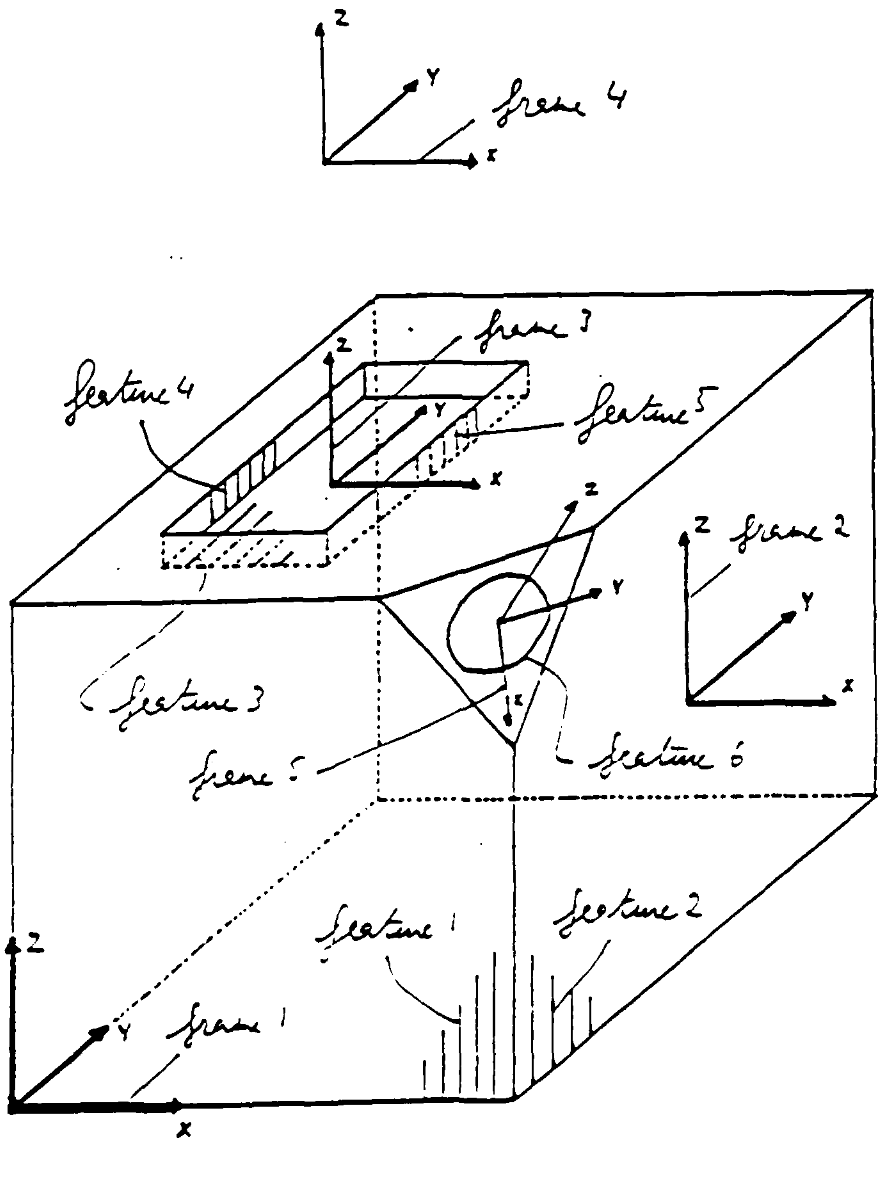
The global procedure for creating a robot program, including calibration of the world model was briefly outlined in the introduction. For a better understanding of the third step, namely the introduction of features into the world model and the conditioning of the location of important frames upon these features, we will give a short example.

The following figure views an object with 5 frames attached to it. Frame 1 is the natural frame of the object. Frame 2 to 5 are important auxiliary frames necessary in an assembly operation.

To calibrate this model we will introduce six features into the model and condition the location of the frames upon these features according to the following condition types:

- frame 1 will be connected to features 1 to 5, which impose condition C1 on the frame
- frame 2 will be connected to feature 2, which imposes condition C12 on the frame
- frame 3 will be connected to feature 3, which imposes condition C6 on the frame
- frame 4 will not be connected to a feature, but it is connected to frame 3
- frame 5 will be connected to feature 6, which imposes condition 13 on the frame

After measuring the 6 features the location of the 5 frames will be updated according to the imposed conditions.



Appendix 1

If a point $\underline{v} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ lies in a plane $\Re = [a\ b\ c\ d]$ then the matrix product:

$$\Re \cdot \underline{v} = 0 \tag{1}$$

or in expanded form:

$$a \cdot x + b \cdot y + c \cdot z + d = 0 \tag{2}$$

Let $\underline{T}$ be a transformation matrix representing a natural frame (X', Y', Z') of an object relative to the world frame (X, Y, Z).

Given $[x'\ y'\ z'\ 1]^T$ the coordinates of a point relative to the natural frame and $[x\ y\ z\ 1]^T$ the coordinates of that same point relative to the world frame, then:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \underline{T} \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \tag{3}$$

Given [a' b' c' d'] the row matrix of a plane relative to the natural frame and [a b c d] the row matrix of the same plane relative to the world frame, then:

$$[a\ b\ c\ d] = [a'\ b'\ c'\ d'] \cdot \underline{T}^{1} \tag{4}$$

as we require that:

$$[a\ b\ c\ d] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [a'\ b'\ c'\ d'] \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \tag{5}$$

To verify this we substitute from (3) and (4) into the left hand side of (5) and we obtain:

$$[a'\ b'\ c'\ d'] \cdot \underline{T}^{1} \cdot \underline{T} \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = [a'\ b'\ c'\ d'] \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

If we have more than one plane, we can easily expand (4) into:

$$\begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ \dots & & & \\ a_n & b_n & c_n & d_n \end{bmatrix} = \begin{bmatrix} a_1' & b_1' & c_1' & d_1' \\ a_2' & b_2' & c_2' & d_2' \\ \dots & & & \\ a_n' & b_n' & c_n' & d_n' \end{bmatrix} \cdot \underline{T}^{-1}$$