# Task Frames in Robot Manipulation

Dana H. Ballard
Department of Computer Science
University of Rochester
Rochester, NY 14627

## Abstract

Most robotics computations refer to a single world-based frame of reference; however, several advantages accrue with the introduction of a second frame, termed a task frame. A task frame is a coordinate frame that can be attached to different objects that are to be manipulated. The task frame is related to the world-based coordinate frame by a simple geometric transformation. The virtues of such a frame are: (1) certain actions that are difficult to specify in the world frame are easily expressed in the task frame; (2) the task-frame to task-world transformation provides a formalism for describing physical actions; and (3) the task frame can be related to the world frame by proprioception.

------------------------

*This research was supported in part by the National Science Foundation under Grant MCS-8203920.

## 1. Introduction

Robotics problems are best considered at different levels of abstraction. This is because many problems can be analyzed effectively within a given abstraction level without appealing to other levels. For example, in robot planning it is often helpful to consider actions symbolically without involving details of the servomechanisms that implement such actions. The standard symbolic description for the action *move x from y to z* in a STRIPS-like expression [Fikes and Nilsson, 1971] is:

MOVE(x, y, z)
    Preconditions: CLEAR(x); CLEAR(z); On(x,y)
    Postconditions: ON(x,z); CLEAR(y).

The principal advantage of such a system is that symbolic plans involving several actions which achieve a set of goal conditions can be created systematically. The disadvantage of this level is that important geometric details are suppressed. For example, the predicate CLEAR(x) may depend on the geometry of the environment and the manipulator. Objects that might be CLEAR with respect to a multiple degree-of-freedom manipulator might not be CLEAR to a low-degree-of-freedom cartesian manipulator.

Similar kinds of arguments can be made for the lowest level of abstraction, the servomechanism itself. The basic problem of the servomechanism is to exert forces on objects in the world and transport them along desired trajectories. The control problem of "given a trajectory, find the actuator torques required to follow it," can be solved independently from the symbolic plan using only the inverse dynamics of the manipulator. At the symbolic level, problems can be solved independent of the details; just as important, at the servomechanism level problems can be solved independent of the context.

The natural level of abstraction to introduce between the symbolic level and servo level is a geometric level. The geometric level provides an explicit representation of space that includes geometrical and mechanical structure. For example, while CLEAR(x) may be a necessary property for an action at the symbolic level, the geometrical level contains the necessary structure that allows this property to be established. Another element of the geometrical level is that it must be able to communicate with the servomechanism level. More specifically, we argue that the geometrical level must contain structure that functions as a command language for the servomechanism level. To first order, we argue that manipulators need only these three levels, which are described in Table 1.

**Table 1: Levels of Abstraction in Robot Manipulation**

| Level | Description |
|---|---|
| symbolic | STRIPS-like description of actions; planning done by chaining appropriate actions to change current state to goal state |
| geometric | representation of space in which actions take place; command language for servomechanism level |
| servomechanism | detailed description of manipulator: inertial parameters, friction models, manipulator internal geometry |

The central idea of this paper is that of a *task frame*. A task frame is described at the geometric level and is a geometric coordinate frame that is attached to the object being manipulated. To understand the notation for a task frame, one must appreciate that current robot manipulation and control strategies tend to refer computations to a single reference frame, termed the *world frame*. An example of such a strategy is that of compliance [Paul, 1981], where the manipulator is constrained to move along given geometric surfaces. Such surfaces have been termed *C-surfaces* [Mason, 1981]. For example, when turning a crank, the crank handle will traverse a given path in the world frame, as shown by Figure 1a, which is taken from [Brady et al., 1982].

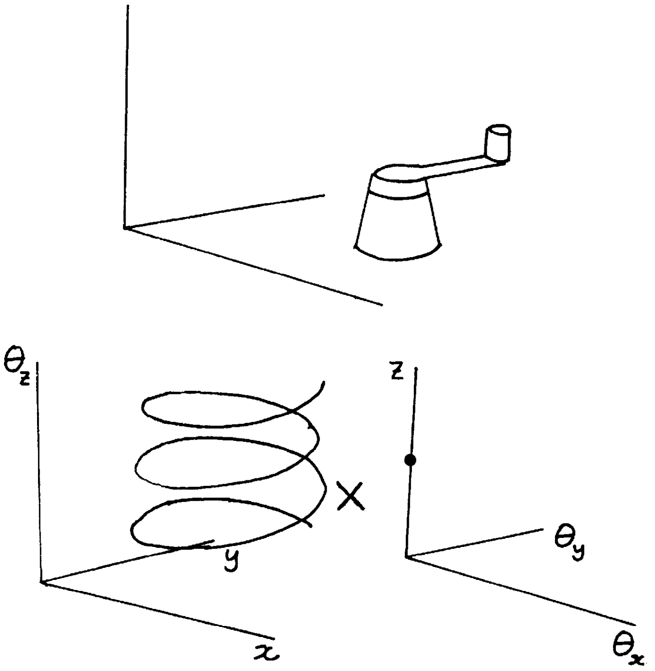
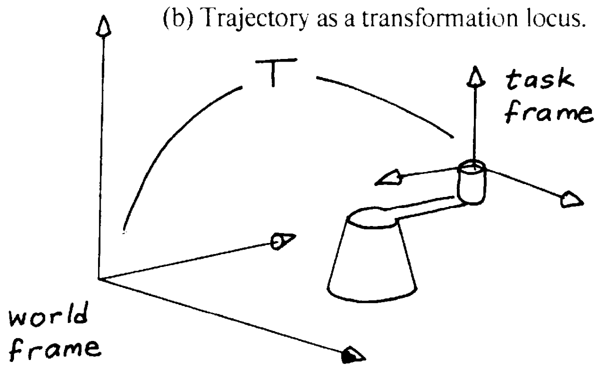


Figure 1: (a) Trajectory as a world-frame locus.

(b) Trajectory as a transformation locus.



A *task frame* is related to *C* surfaces but is different in a crucial way. Rather than thinking about the *C*-surface in the world frame, the task frame is intrinsically fixed to the object being manipulated (for the duration of the task) and is related to the world frame by the obvious geometric transformation. That is, given any two of the set (task frame, world frame, task-world transformation), the third is easily computed. Figure 1b shows the characterization of the task frame for the problem of turning the crank. The two representations are equivalent in the sense that either one could be transformed into the other. The important difference is that within the task frame formalism, the problem of turning the crank can be simply described as "push in the $e_1$-direction until you meet resistence." The expedient of introducing an intrinsic frame and separating the intrinsic frame from its transformation has the following advantages:

1) many actions that are difficult to express in the world frame have very simple expressions with respect to the task frame and transformation (in fact, they can be described by invariants with respect to these two entities);
2) the frame-transformation decoupling allows us to relate simply geometric and mechanical changes in the world with corresponding symbolic descriptions; and
3) the transformation between the task frame and world frame can be computed via both vision and proprioception.

The crux of the rest of the paper is to describe these advantages in detail. Section 2 contains the basic notions from geometry and mechanics needed to understand the subsequent material. Section 3 describes the interface between the geometric and symbolic levels. The focus is on the recognition and implementation of actions that involve mechanics. Section 4 describes the interface between the geometric and servomechanism levels. The focus is on the self-calibration necessary to relate the task and world frames, and the automatic relations between the two levels introduced by the task frame formalism.

## 2. Geometry and Mechanics

### A. Geometry

An orthogonal geometric coordinate frame consists of three vectors, $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, such that any two pair are mutually perpendicular ($\mathbf{e}_i \cdot \mathbf{e}_j = 0$, for $i \neq j$), they form a right-handed coordinate system ($\mathbf{e}_3 = \mathbf{e}_1 \times \mathbf{e}_2$), and all the vectors are unit vectors ($\mathbf{e}_i \cdot \mathbf{e}_i = 1$). To denote the task frame the subscript t is used, i.e., $\mathbf{e}_{1t}$, $\mathbf{e}_{2t}$, $\mathbf{e}_{3t}$, and to denote the world frame the subscript w is used. Each frame has an *origin* $\mathbf{x} = (x, y, z)$, so that $\mathbf{x}_t$ is the origin of the task frame and $\mathbf{x}_w$ is the origin of the world frame. The world frame can be thought of as described in terms of master coordinates. In this case $\mathbf{x}_w = 0$, $\mathbf{e}_{1w} = (1, 0, 0)$, $\mathbf{e}_{2w} = (0, 1, 0)$, and $\mathbf{e}_{3w} = (0, 0, 1)$. To denote a frame $(\mathbf{x}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ we use E.

Given the task frame and world frame, the transformation between them can be specified by a rotation and a translation. The understanding is that the rotation is done first since rotation and translation do not commute. The transformation is specified by an origin change $\Delta\mathbf{x} = (\Delta x, \Delta y, \Delta z)$ and a rotation $(\mathbf{n}, \theta)$. The later notation stands for a rotation $\theta$ about a unit vector $\mathbf{n}$ where n is expressed in world coordinates. The transformation can be computed directly as:

$$\Delta\mathbf{x} = \mathbf{x}_t - \mathbf{x}_w$$

and, assuming a quaternion representation for rotations [Pervin and Webb, 1983]:

$$\mathbf{n} = \text{Normalize}((\mathbf{e}_{1t} - \mathbf{e}_{1w}) \times (\mathbf{e}_{2t} - \mathbf{e}_{2w}))$$

$$\theta = (-(\mathbf{n} \times \mathbf{e}_{1t})(\mathbf{n} \times \mathbf{e}_{1w}))^{1/2}$$

### B. Mechanics

A robot manipulator is a series of links. Each link is independently controlled by its own servomotor. The links can be described by joint angles $\underline{\theta}$ (rotary joints) which are controlled by applying torques $\underline{\tau}$. One such configuration is the two-link, planar manipulator suggested by [Horn, 1975]. Figure 2 shows the manipulator geometry. The force and torque applied at the tip of the manipulator can be described by a vector$(\mathbf{f}, \mathbf{n})$. The external force and torque can be related to the joint torques by general dynamic equations

$$F(\underline{\tau}, \underline{\theta}, \mathbf{f}, \mathbf{n}) = 0 \qquad (1)$$

To drive the arm, $\underline{\theta}$, $\mathbf{f}$, $\mathbf{n}$ are assumed known and Equation 1 is solved for the control torques $\underline{\tau}$. This way of solving (1) is known as the inverse dynamics. That is, the torques $\underline{\tau}$ can be related to $(\mathbf{f}, \mathbf{n})$ by a set of equations: $\underline{\tau} = f^{-1}(\underline{\theta}, \mathbf{f}, \mathbf{n})$. Recently developed solution techniques have made it practical to solve the inverse dynamics equations in real time [Luh et al., 1980; Hollerbach, 1980]. The easier problem is readily also solved; that is, given $\underline{\tau}$, $\mathbf{f}$, and $\mathbf{n}$, determine $\underline{\theta}$.

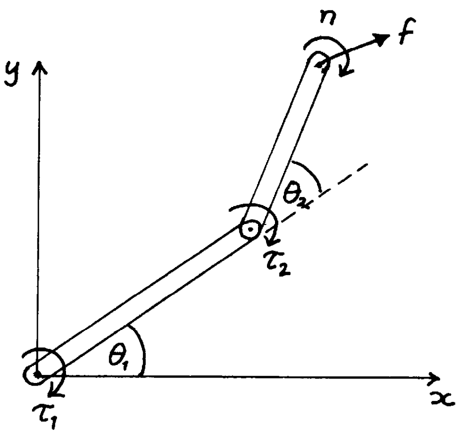


Figure 2: Two-link planar manipulator.

Besides the dynamics problem there is the *kinematics* problem. Given the state of the system in terms of $\underline{\theta}$, $d\underline{\theta}/dt$, $d^2\underline{\theta}/dt^2$, one must determine the motion of the manipulator tip $x_p$, $dx_p/dt$, $d^2x_p/dt^2$. This is the easy part. The reverse problem--given $\mathbf{x}$ find $\underline{\theta}$--is harder but can be solved analytically by designing manipulators with special geometries. One such geometry is a spherical wrist [Featherstone, 1983].

In very simple manipulator geometries both the inverse dynamics and inverse kinematics may have analytical solutions. For example, in the two-link planar manipulator, the joint torques $\tau_1$ and $\tau_2$ may be expressed as

$$\tau_1 = A + B + Cf_x + Df_y + n$$

$$\tau_2 = E + F + Gf_x + Hf_y + n \qquad (2)$$

where A, B, C, D, E, F, G, and H are expressions involving the manipulator joint angles $\underline{\theta}$, joint angular velocities and accelerations, mass, and inertial parameters. The letters A and E denote terms dependent on velocity and acceleration; the letters B and F denote terms dependent on gravity. These equations show that the problem of controlling (**n**, **f**) in this case is underdetermined. However, if the problem is simplified slightly, e.g., the gripper is a finger such that $\mathbf{n} = \mathbf{0}$, then the external forces at the tip can be directly related to the control torques. Our example assumes such a gripper.

## 3. Geometry and Symbols: Recognizing and Implementing Actions

This section shows how the geometrical notions of a task frame and transform can be of general use in a symbolic planner. In particular: (1) the framework allows the interrelation of symbolic and geometric descriptions of actions; (2) the task frame and transformation allows a simple description of tasks in terms of invariants; and (3) the task frame allows checking for collisions between objects.

To start with the first point, consider the description of *falling*. If an object is falling then its origin is approaching that of the world frame origin such that the z-velocity is negative. In other words,

FALLING(obj) $\Longleftrightarrow$ $(v_z < 0)$

where the understanding is that expressions involving positions and orientations and changes in such are *statements about the world-frame to task-frame transformation*. Expressing the process of *falling* as a rate of change has the effect of making it comparable to a static situation. The logical expression $(v_z < 0)$ must hold throughout the *falling* process. The task frame orientation may also play a role in the description. For example:

RIGHT-SIDE-UP(obj) $\Longleftrightarrow$ ALIGNED($e_{3t}$, $e_{3w}$)

In this as in the previous example, a first order logic syntax is assumed with expressions consisting of predicates denoted by upper case, terms denoted by lower case. The point of these examples is that for problems involving Newtonian mechanics, the task-frame to world-frame transformation provides the basis for systematically relating symbolic expressions and geometrical expressions.

The task frame structure leads naturally to a formalism at the geometric level for describing actions. This formalism has three principal advantages: (1) its elements are all invariants; (2) the formalism is sufficiently abstract that the same action can be used in a variety of contexts; and (3) its structure can be interpreted by the servomechanism.

The format that we adopt for representing actions has a STRIPS-like syntax. Each action has a set of preconditions that must be true for the action to be applicable, a set of while conditions that must hold during the execution of the action, and a set of stopping conditions. Thus an action is described as:

```
ActionName(params)
    if {preconditions} then do {whileconditions}
    until {stoppingconditions}
```

where the parameters are used by the various conditions.

This structure takes advantage of the previous development which related symbolic constraints and geometric constraints. Consider the example of closing a door. This can be expressed symbolically as:

```
DOORCLOSING(door)
    if TOUCHING(door) then do PUSH(door)
    until ARRESTED(door)
```

but also can be expressed geometrically as

```
DOORCLOSING(door)
    if $(\epsilon,0,0)$ then do $(f,0,0)$ & ALIGNED($E_t$,$E_{handle}$)
    until $(0,0,0)$
```

This syntax illustrates a number of important points which we will now elaborate. In the first place, note that we have been able to decouple the force constraints from the geometric constraints. (A similar decoupling is seen in *C*-surfaces for the cases of pure force or pure position control [Mason, 1981].) Thus rather than specify the force control of the handle in the world frame [Mason, 1981] where it has a varying locus, it is specified in the task frame where it has a very simple structure. The notation $\mathbf{f}_t = (f_1, f_2, f_3)$ stands for *exert force* $\mathbf{f}_t$ *in the task frame coordinate system*. The quantity $\epsilon$ is a small contact force, less than that required to move the object, whereas f is large enough to start the object moving. The key virtue of the task frame is that the force $\mathbf{f}_t$ can be an invariant during the action.

The second part of the while condition for closing the door expresses the relationship between the task frame and some other frame expressed in world frame coordinates. For the earlier discussion one can appreciate that given $E_t$ and $E_{handle}$, the predicate ALIGNED($E_t$,$E_h$) is easy to compute.

The following scenario is imagined for the action DOORCLOSING. Given that the handle is grasped, the servomechanism applies a force in the $\mathbf{e}_{1t}$ direction of the task frame to move the door. The door moves until it bumps into the door frame, at which time the frame exerts a force **f** to cancel the manipulator torques. This satisfies the stopping condition. Details such as the microdynamics of the contact are left to the servomechanism, and we will defer the discussion of these details until the next section.

The above strategy does not address the problem of slamming the door. This can happen when the force is too large. To deal with this example we will add one condition and change notation slightly. First we add the while condition ($\|\mathbf{v}\| \in [\nu_0 - \Delta\nu, \nu_0 + \Delta\nu]$). This states that the speed of the task frame with respect to the world frame is to be constrained in the interval $\nu_0 \pm \Delta\nu$. As a shorthand, we use capital letters to specify intervals, i.e., $\nu_0 \pm \Delta\nu \equiv V_0$. The second change we will make is to relax the force specification in the task frame to just the specification of the axis to be controlled, in this case $\mathbf{e}_1$. The while condition becomes:

$$\begin{aligned}\{\text{whilecondition}\} = \;& \text{FORCECONTROL}(\mathbf{e}_{t1}) \\ & \textbf{and } (\|\mathbf{v}\| \in V_0) \\ & \textbf{and } \text{ALIGNED}(E_t,E_h)\end{aligned}$$

The understanding is that the servomechanism can use this to generate the appropriate commands. One simplistic possibility is:

$$\begin{aligned}&\text{if } \|\mathbf{v}\| > \nu_0 + \Delta\nu \quad \text{then} \quad f_1 := f_1 + \Delta \\ &\text{if } \|\mathbf{v}\| < \nu_0 + \Delta\nu \quad \text{then} \quad f_1 := f_1 - \Delta\end{aligned}$$

Notice that although the while conditions have become more complex, their essential structure has been maintained in that they are invariants with respect to the action.

We now turn to the second advantage of the formalism, which is that, once appropriate bindings have been established, a wide variety of different situations can be described by the same task frame description of the action. Figure 3 shows two different tasks which can be handled by DOORCLOSING. In the first, gravity is assumed to be perpendicular to the plane, i.e., the figure shows a top view.

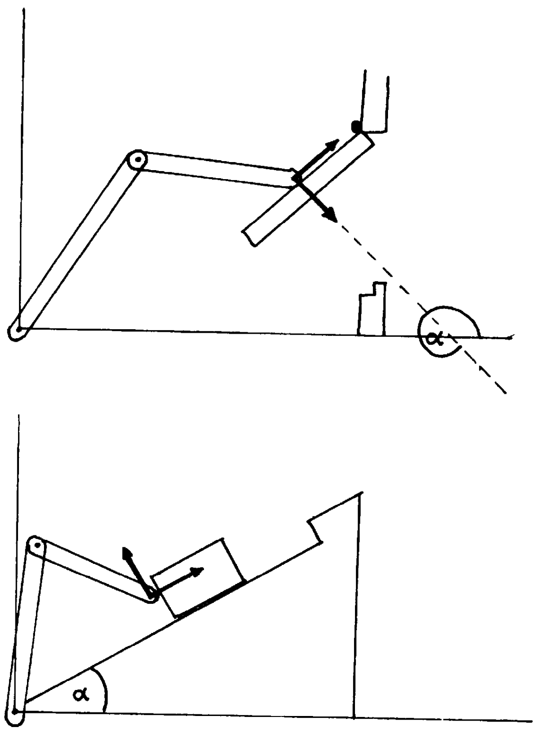


Figure 3: Different tasks which can be handled by DOORCLOSING.

In these examples, recognizing that the described action is one of DOORCLOSING from the geometric features would be difficult but perhaps not impossible. More plausibly, the relevant geometric features of the problem, which in this case are specified by $E_t$, may already be known. In any case, the geometric level is the essential starting point from which the relevant constraints can be synthesized.

We note that some details are being finessed at this level of description. For example, what if the masses in these examples are such that the servomechanism cannot achieve $\|\mathbf{v}\| \in V_0$? This case of failure has to be resolved at the planning level, and aside from characterizations of the failure mode, we are not addressing these kinds of problems in this paper.

Another problem is that of collision detection. Task frames provide a partial mechanism for handling this problem. First instantiate all the geometrical objects with respect to the task frame, and then use the details of the geometric representation, e.g., constructive solid geometry, to check for solid material from two or more objects occupying the same physical space.

## 4. Geometry and Servomechanisms: Self-Calibration

In order for the task frame scheme to work there must be some way of computing the transformation between the task frame and the world frame. In this section we discuss ways of doing this and show how they can be integrated into the real-time control program of the servomechanism.

One way of establishing the desired transformation is through visual input. This much-researched problem can be done in constant time on a parallel machine if suitable visual features can be identified [Ballard and Sabbah, 1983; Hrechanyk and Ballard, 1983]. But from a robotics context, a more interesting method is to use the inverse dynamics and kinematics of the servomechanism itself. To see how this might work, let us reconsider the problem of closing the door. From the inverse kinematics of the manipulator it is possible to calculate the end effector velocity. In the normal case of door closing, once the door moves, its velocity vector is available in world-frame coordinates. In this case of compliant motion, *the door can only move in the direction of the* $e_1$ *axis of the task frame.* Thus, $\mathbf{e}_1$ can be computed as $\mathbf{e}_1$ = Normalize(**v**), where **v** can be measured from the kinematic equations. Since $\mathbf{e}_1$ is the crucial axis in the closing task, the other axes may not need to be updated beyond enforcing the orthogonality condition. If they should be updated, an additional constraint is that, for two different times $t_1$ and $t_2$, then $e_3$ can be computed as $e_3$ = Normalize($\mathbf{v}(t_1) \times \mathbf{v}(t_2)$), where the two velocities must have different directions. This is a natural constraint in the door closing situation. In pushing the block the velocity needs to wobble arbitrarily while being pushed to establish the second direction vector.

Another way to calculate the transform is via force proprioception. Assuming the velocity parallel to the door surface is clamped at zero, the transformation parameter, which in 2D is a single angle (denoted by $\alpha$ in Figure 3), can be readily computed. The transform is assumed to be initialized at the beginning of the action and continuously updated during the action. One way of updating is:

1) use $\underline{\tau}(t)$, **f**(t) to solve for $\underline{\theta}(t)$; (3)
2) use $\underline{\theta}(t)$ to solve for $x_p(t)$;
3) $\alpha' = \tan^{-1}((dy_p/dt)/(dx_p/dt))$;
4) **if** $|\alpha - \alpha'| < \epsilon_1$, **then** $\alpha := \alpha'$;
   **else** *failure*.

Now we turn to the stopping condition. If stopped, B = F = 0 in (2). This leads to:

1) compute $\underline{\tau}_s$ using B = F = 0 and f' measured from proprioception;
2) if $|\underline{\tau} - \underline{\tau}_s| < \epsilon_2$, then stop.

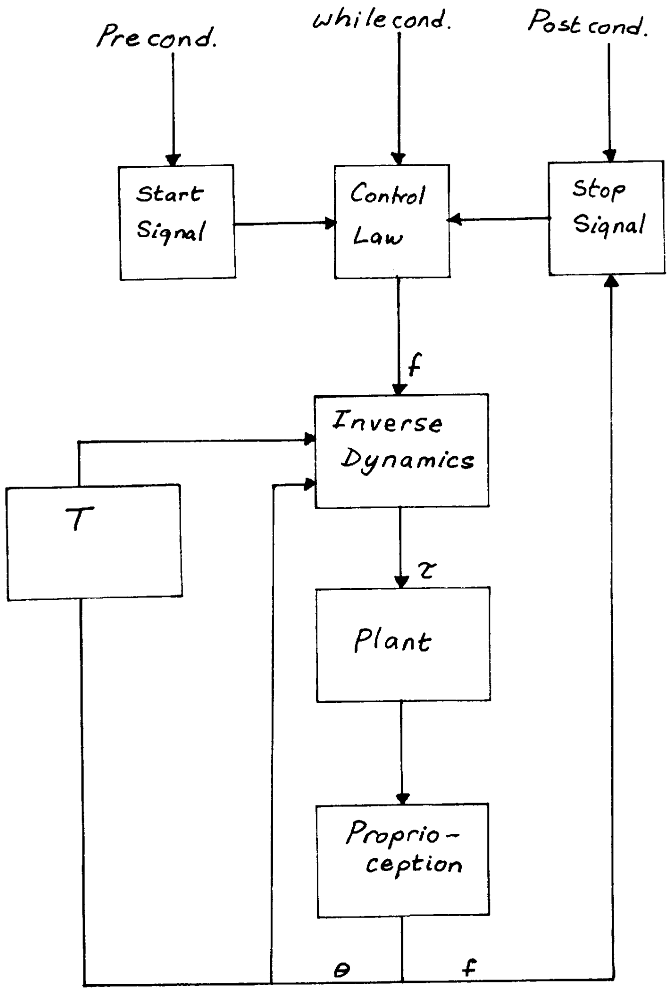


Figure 4: Details of servomechanism level.

These kinds of computations can be utilized by a servo controller in the manner depicted in Figure 4. To see how the controller works, consider again the door closing action. First, at the symbolic level, the symbolic description of the action can be automatically translated into task-frame constraints. Second, actions at the geometric level can be automatically translated into servomechanism commands: the *if* conditions are utilized to generate a start signal, the *while* conditions are utilized to synthesize a control function, and the *until* conditions are utilized in a *termination monitor*. In other words, the invariants at the geometric level become set points for the controller at the servomechanism level. Third, at the servo level, the controller computes a command signal in the task frame. This command is translated into world-frame coordinates by the task-world transformation. The inverse dynamics allows the actuator torques to be synthesized from the desired control signal. The actuator torques have an effect on the plant which is monitored by proprioception. Proprioception uses the inverse dynamics but assumes the torques and system state are known in order to estimate the world forces and velocities. These are checked against the termination conditions and also used to update the task-world transformation. The termination condition is propagated to the symbolic level where ARRESTED(door) is set to TRUE.

## 5. Summary

Task frames make many issues that arise in robot planning and manipulation simpler. The change from earlier work has been inspired in part by recent work at the servomechanism level which has allowed the development of dynamically accurate plant models [Mukerjee et al., 1984] and aforementioned fast solutions to the problems of inverse kinematics and dynamics. This means that the main portion of manipulator control can be carried out as an open loop rather than a closed loop. Before these advances, manipulator control has had to be segregated into a planning phase and an acting phase, and the dynamics of the acting phase could not be introspected during the planning phase. With accurate plant models and open loop control strategies, the planning and acting phases can be more intimately linked.

It is important to acknowledge that this paper does not tackle many issues that must be solved to make robot manipulation practical. Some of these are: trajectory planning, recovering from failures, and the representation of large amounts of detailed spatial information. The exposition is limited to characterizing single actions and showing how they may be characterized as geometrical and mechanical invariants. Hopefully this representational strategy will make the solution of the other problems easier.

## 6. References

Ballard, D.H. and D. Sabbah, "Viewer independent shape recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence 5*, 6, November 1983.

Brady, M., J.M. Hollerbach, T.L. Johnson, T. Lozano-Perez, and M.T. Mason. *Robot Motion: Planning and Control.* Cambridge, MA: The MIT Press, 1982.

Featherstone, R., "Position and velocity transformations between robot end effector coordinates and joint angles," *Int. J. Robotics Research 2*, 1983.

Fikes, R.E. and N.J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence 12*, 3/4, 184-209, 1971.

Hollerbach, J.M., "A recursive formulation of Lagrangian manipulator dynamics," *IEEE Trans. Systems, Man, Cybernetics SMC-10*, 11, 730-736, 1980.

Horn, B.K.P., "Kinematics, statics, and dynamics of two-d manipulators," MIT AI Lab, Working Paper 99, June 1975.

Hrechanyk, L.M. and D.H. Ballard, "A connectionist model for shape perception," Computer Vision Workshop, Ringe, NH, August 1982; also appeared as "Viewframes: A connectionist model of form perception," DARPA Image Understanding Workshop, Washington, D.C., June 1983.

Luh, J.Y.S., M.W. Walker, and R.P.C. Paul, "On-line computational scheme for mechanical manipulators," *J. Dynamic Systems, Measurement, Control 102*, 69-76, 1980.

Mason, M.T., "Compliance and force control for computer controlled manipulators," *IEEE Trans. Systems, Man, and Cybernetics 11*, 6, 418-432, 1981.

Mukerjee, A., R.C. Benson, and D.H. Ballard, "Towards self-calibration in robot manipulator systems: Dynamics enhancement through trajectory deviation analysis," Working Paper, Depts. of Mechanical Engineering and Computer Science, U. Rochester, March 1984.

Paul, R.P. *Robot Manipulators: Mathematics, Programming, and Control.* Cambridge, MA: MIT Press, 1981.

Pervin, E. and J.A. Webb, "Quaternions in computer vision and robotics," *Proc.*, IEEE Computer Vision and Pattern Recognition Conf., 382-383, Washington, DC, June 1983.