Proceedings of DETC'04
ASME 2004 Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
September 28-October 2, 2004, Salt Lake City, Utah, USA

DETC2004-57447

MANIPULATOR TASK-BASED PERFORMANCE OPTIMIZATION

Chalongrath Pholsiri
Research Assistant
Robotics Research Group
Department of Mechanical Engineering
The University of Texas at Austin
longrath@mail.utexas.edu

Chetan Kapoor
Chief Scientist
Robotics Research Group
Department of Mechanical Engineering
The University of Texas at Austin
chetan@mail.utexas.edu

Delbert Tesar
Professor, Director
Robotics Research Group
Department of Mechanical Engineering
The University of Texas at Austin
tesar@mail.utexas.edu

ABSTRACT

This research uses new developments in redundancy resolution and real-time capability analysis to improve the ability of an articulated arm to satisfy task constraints. Task constraints are specified using numerical values of position, velocity, force, and accuracy. Inherent in the definition of task constraints is the number of output constraints that the system needs to satisfy. The relationship of this with the input space (degrees of freedom) defines the ability to optimize manipulator performance. This is done through a Task-Based Redundancy Resolution (TBRR) scheme that uses the extra resources to find a solution that avoids system constraints (joint limits, singularities, etc.) and satisfies task constraints. To avoid system constraints, we use well-understood criteria associated with the constraints. For task requirements, the robot capabilities are estimated based on kinematic and dynamic manipulability analyses. We then compare the robot capabilities with the user-specified requirement values. This eliminates a confusing chore of selecting a proper set of performance criteria for a task at hand. The breakthrough of this approach lies in the fact that it continuously evaluates the relationship between task constraints and system resources, and when possible, improves system performance. This makes it equally applicable to redundant and non-redundant systems. The scheme is implemented using an object-oriented operational software framework and its effectiveness is demonstrated in computer simulations of a 10-DOF manipulator.

1. INTRODUCTION

Kinematic redundancy gives robotic manipulators human arm-like versatility. This allows redundant robots to perform complex tasks that are otherwise impossible. With versatility, however, comes complexity of how to control the robot. Redundancy provides the robot with multiple choices from which a decision needs to be made in an intelligent manner as to finding the best solution. The process of solving the inverse kinematics problem of redundant robots is called *redundancy resolution* and methods for solving this problem are termed *Redundancy Resolution Techniques* (RRTs).

Researchers have attempted to solve the redundancy problem for decades. Generalized inverses yield solutions that minimize the two-norm of the joint velocity vector or minimize the joint kinetic energy [19]. Liegeois [12] utilized redundancy to avoid joint limits by the gradient projection method. Redundancy is also used in avoiding obstacles [7], avoiding singularities [20], torque minimization [8], and so on.

The notion of *task oriented* optimization for redundant manipulators is presented in [5] based on the task ellipsoid concept and robot's velocity and force transmission ratios. The idea is to minimize a task-dependent performance measure defined as the deviation of the transmissions ratios along the task directions.

Due to the realization that optimizing only a single criterion may not be sufficient in most tasks, the focus has been shifted to Multi-criteria Redundancy Resolution Problem (MRRP), where

redundancy is used to achieve *multiple performance objectives*. Perhaps the most common technique of MRRP is to linearly combine all the criteria into a composite performance index. Criteria are given weights according to their perceived importance. The composite index is then used in such optimization schemes as the gradient projection method [12] or compact quadratic programming method [2].

Due to the popularity of the weighted sum method, a great deal of attention has been paid to devising the best weighting scheme. Several weighting strategies have been proposed, including a probability-based weighting technique [13], a fuzzy-logic supervisor [6], and the parallel scheme [3].

Avoiding the weighted sum method, Pamanes and Zeghloul [16] performed the optimization of multiple criteria by assigning single performance measures to pre-specified points along the trajectory. Another approach called *task prioritization* divides a task into subtasks of different degrees of priority. Subtasks with higher priority are to be performed first and less important subtasks are satisfied only in the null space of the higher-priority tasks [14]. The task prioritization approach is also formulated in a recursive form to allow easy integration of any number of subtasks [18].

The major problem with traditional multi-criteria RRTs, is that almost all of them require that the user define a set of criteria and their relationships for a given task. It is not straightforward to come up with a proper set of criteria, not to mention the relationships of one criterion to one another (i.e. which one is more important and by how much) because it is not obvious what kind of impact these criteria will have on the task. The main reasons are some of these criteria lack clear physical meanings and many of the criteria are coupled.

For example, a velocity transmission ratio is a ratio between the magnitude of the EEF velocity in the direction of interest and the norm of the joint velocity vector. Maximizing this criterion means the EEF can move in the desired direction with minimal effort from the actuators. At the same time, maximizing velocity transmission ratio also means that the effect of joint errors is also maximized at the EFF, which is certainly not desirable. Generalized stiffness is the overall ability of the EEF to withstand a general load without causing too large a deflection. Maximizing this criterion can reduce the deflection at the EEF and therefore improve accuracy. Some questions arise. Should we minimize or maximize the velocity transmission ratio criterion? Under what circumstances should we maximize the generalized stiffness criterion and not the velocity transmission ratio or vice versa? Or we probably wish to maximize both of these criteria but which one should we give more import? Suppose the operator wants the robot to be able to exert maximum force in the Y direction. He can certainly try to maximize the force transmission ratio in the Y direction. However, what if the operator wants the robot to be able to exert maximum force in the Y direction and in the mean time achieve a certain level of accuracy tracking in the X direction? He could try to minimize the velocity transmission ratio in the X direction while maximizing the force transmission ratio in the Y direction. However, what value of weight should be assigned to each criterion? What if the accuracy needs to be maintained in the Z direction too? To the best of the author's knowledge, no work has been done that could address this problem.

In this paper, we present an alternative approach called *Task-Based Redundancy Resolution* (TBRR). Instead of optimizing multiple performance criteria in a conventional fashion, TBRR searches the null space for solutions that satisfy all the ***system constraints and task requirements in terms of speed, force, and accuracy*** to ensure that the task at hand can be carried out successfully and satisfactorily. This eliminates the perplexing process of selecting and combining multiple criteria that usually exists in traditional RRTs.

2. ROBOTICS TASKS

Because the whole purpose of TBRR is to exploit redundancy in such a way that tasks can be successfully executed, it is first mandatory to discuss what robotic tasks are in the context of this work. Robotic tasks can be described at many levels of abstraction. At the highest level, robots receive human-like commands such as "open the door" or "paint the panel" and then proceeds to execute the task. In a more common fashion, numerical task descriptions are given to the robot and the robot follows these commands. Most robotic tasks can be represented by the numerical descriptions of the end-effector (EEF) path, speed, force, and accuracy. Robotic task executions are considered successful if no constraint is violated and all task requirements are satisfied.

2.1 System Constraints

System constraints are physical or operational limitations that may prevent the robot from successfully executing the task. Several types of constraints exist. Among them are joint travel limits, joint torque limits, obstacles, singularities, etc. Not only does violating these constraints guarantee a failure in task execution, but it could also lead to environmental or system damage. These constraints usually have (constraint-based) criteria associated with them. One common characteristic of constraint-based criteria is that they generally have definite physical meanings that can easily be interpreted. Two types of constraints are discussed here.

Joint Travel Limit

Joint travel limits are mechanical limits on the robot joints. During operation, all joints must be maintained within these limits. The Joint Range Availability (JRA) criterion is defined as

$$\gamma_{JRA} = \min_i \left(1 - \frac{\left| \theta_i - \theta_{i,\text{mid}} \right|}{\theta_{i,\max}} \right) \qquad (1)$$

where θ_i is the joint displacement, $\theta_{i,\text{mid}}$ the displacement at the midpoint of the travel range, $\theta_{i,\max}$ the displacement at the travel limits of joint *i*. JRA=1 signifies the best possible configuration where all joints are in the middle of their ranges. JRA=0 indicates that at least one joint is at its limit. It is necessary that the JRA criterion be kept above 0, preferably in the range of 0.05-0.1 throughout the operation.

Mathematical Singularity

Several inverse kinematics schemes fail when the Jacobian matrix loses its rank at mathematical singularities. Therefore, it is crucial that singularities be avoided. Perhaps the most common criterion used in singularity avoidance is the Measure of Manipulability (MOM) defined as

$$\gamma_{MOM} = \sqrt{\det(JJ^T)} \tag{2}$$

where J is the Jacobian matrix [20]. MOM approaches zero at singularities. In reality, however, if MOM falls below a positive threshold whose value is robot-dependent (it is even unit-dependent), it could cause errors in the inverse kinematics scheme. Therefore, it is necessary to keep MOM above the threshold. It should be noted that MOM has multiple physical interpretations, some of which are unclear. Many studies, including [16] and [20], used MOM as one of the performance objectives for maximization, which may not yield the best solution for a given task. MOM is used here strictly to avoid mathematical singularities.

2.2 Task Requirements

For a robotic task to be performed successfully, we must satisfy all the task requirements, which are defined in terms of desired speed, force, and accuracy at the EEF. It is therefore necessary to be able to estimate the robot's *achievable* speed, force, and accuracy and compare them to the desired values. This section discusses how to estimate these robot capabilities from the robot properties, actuator capacities, and robot configuration.

Achievable EEF Speed

The robot's EEF velocity is related to the joint velocity by

$$\dot{x} = J\dot{\theta} \tag{3}$$

where $\dot{x} \in \mathcal{R}^m$ is the EEF velocity and $\dot{\theta} \in \mathcal{R}^n$ is the joint velocity vector. A normalized joint velocity vector is defined as

$$\tilde{\dot{\theta}} = L_{\dot{\theta}}^{-1}\dot{\theta}, \tag{4}$$

where $L_{\dot{\theta}} = diag\left\{\dot{\theta}_1^{\max}, \dot{\theta}_2^{\max}, \ldots, \dot{\theta}_n^{\max}\right\}$ is an $n \times n$ diagonal matrix consisting of the joint speed limits. It is assumed here that the joint speed upper and lower limits are of equal magnitude but in the opposite direction. It should be noted here that the use of the normalized joint velocity vector allows the following formulations to be valid even with the presence of both prismatic and revolute joints. Substituting $\dot{\theta}$ from (4) into (3) yields

$$\dot{x} = JL_{\dot{\theta}}\tilde{\dot{\theta}}. \tag{5}$$

The EEF velocity is limited by the velocity ellipsoid

$$\left\|\tilde{\dot{\theta}}\right\|_2^2 = \tilde{\dot{\theta}}^T\tilde{\dot{\theta}} = \dot{x}^T\left(JL_{\dot{\theta}}^2J^T\right)^{-1}\dot{x} \le 1. \tag{6}$$

Let $\dot{x} = v\hat{t}$ where $\hat{t}$ is the unit vector in the direction of interest in the task space. Then, (6) becomes

$$\left[\hat{t}^T\left(JL_{\dot{\theta}}^2J^T\right)^{-1}\hat{t}\right]v^2 \le 1. \tag{7}$$

The maximum achievable EEF speed in the $\hat{t}$ direction is

$$v_{\max} = \pm\frac{1}{\sqrt{\hat{t}^T\left(JL_{\dot{\theta}}^2J^T\right)^{-1}\hat{t}}}. \tag{8}$$

Achievable EEF Acceleration

Note that section above describes the kinematic constraint of the achievable EEF speed but does not take into account the dynamics and the torque limits of the robot. These properties dictate how much the robot can accelerate its EEF, which can in turn be a limiting factor of the true EEF speed.

Here we will determine the acceleration capability of a robot with the assumptions that the robot is stationary ($\dot{\theta} = 0$) and the EEF is not constrained. The analytics from this section is borrowed from [4] so the reader is encouraged to consult [4] for detailed derivations of these formulations. The dynamic manipulability ellipsoid of a redundant manipulator can be expressed as

$$\left(\ddot{x} + JM^{-1}g\right)^T\left(JQJ^T\right)^{-1}\left(\ddot{x} + JM^{-1}g\right) \le 1 \tag{9}$$

where M is the $n \times n$ inertia matrix, g is the gravity torque vector, and $Q \equiv ML_\tau^{-2}M$. $L_\tau = diag\left\{\tau_1^{\max}, \tau_2^{\max}, \ldots, \tau_n^{\max}\right\}$ is an $n \times n$ diagonal matrix consisting of the joint torque limits.

Let $N \equiv \left(JQJ^T\right)^{-1}$ and $\ddot{x} = a\hat{t}$ where $\hat{t}$ is the unit vector in the direction of interest in the task space. Then (9) can be rewritten as

$$\alpha f^2 + 2\beta f + \gamma \le 0 \tag{10}$$

where

$$\begin{aligned}\alpha &= \hat{t}^T N\hat{t}\\ \beta &= -\hat{t}^T N\ddot{x}_g\\ \gamma &= \ddot{x}_g^T N\ddot{x}_g - 1\end{aligned}.$$

From (10), if $\beta^2 - \alpha\gamma \ge 0$, the achievable acceleration is in the range of

$$\frac{-\beta - \sqrt{\beta^2 - \alpha\gamma}}{\alpha} \le a \le \frac{-\beta + \sqrt{\beta^2 - \alpha\gamma}}{\alpha}. \tag{11}$$

EEF Position Error

Here we assume that the joint errors are so small that the EEF error vector Δx is related to the joint error vector $\Delta\theta$ through the linear relation $\Delta x = J\Delta\theta$. Similar to the EEF speed, the EEF error is then constrained by

$$\Delta x^T\left(JL_{\Delta\theta}^2J^T\right)^{-1}\Delta x \le 1. \tag{12}$$

where $L_{\Delta\theta} = diag\left\{\Delta\theta_1^{\max}, \Delta\theta_2^{\max}, \ldots, \Delta\theta_n^{\max}\right\}$ is an $n \times n$ diagonal matrix consisting of the maximum joint errors. Similar to the achievable EEF speed in the previous section, the EEF position error can then be estimated as

$$\left|\Delta x\right| \le \frac{1}{\sqrt{\hat{t}^T\left(JL_{\Delta\theta}^2J^T\right)^{-1}\hat{t}}}. \tag{13}$$

Although there are many more types of errors, in this work, we assume that each maximum joint error can be estimated by

combining the joint encoder resolution with the joint deflection due to joint flexibility as shown by

$$\Delta\theta_i^{\max} = \varepsilon_i + [C\tau]_i, \quad i = 1,...,n \tag{14}$$

where ε_i is the encoder resolution of joint i, C is the joint compliance matrix, and τ is the joint torque vector. Naturally, others joint errors can be added to this model.

EEF Static Force Capability

Considering the EEF force and gravity, the joint torques can be expressed as

$$\tau = J^T F + g \,. \tag{15}$$

Define the normalize joint torque vector

$$\tilde{\tau} = L_\tau^{-1}\tau \,. \tag{16}$$

The force ellipsoid can be expressed as

$$\tilde{\tau}^T\tilde{\tau} = \left(J^T F + g\right)^T L_\tau^{-2}\left(J^T F + g\right) \le 1 \,. \tag{17}$$

Let $F = f\hat{t}$ where $\hat{t}$ is the unit vector in the direction of interest in the task space. Then (17) can be rewritten as

$$\alpha_\tau f^2 + \beta_\tau f + \gamma_\tau \le 0 \,, \tag{18}$$

where

$$\alpha_\tau = \hat{t}^T J L_\tau^{-2} J^T \hat{t}$$
$$\beta_\tau = g^T L_\tau^{-2} J^T \hat{t} \,.$$
$$\gamma_\tau = g^T L_\tau^{-2} g - 1$$

Solving (18) yields the robot's static force capability as

$$\frac{-\beta_\tau - \sqrt{\beta_\tau^2 - \alpha_\tau\gamma_\tau}}{\alpha_\tau} \le f \le \frac{-\beta_\tau + \sqrt{\beta_\tau^2 - \alpha_\tau\gamma_\tau}}{\alpha_\tau} \,. \tag{19}$$

3. TASK-BASED REDUNDANCY RESOLUTION

We have defined robotic tasks, system constraints and task requirements. Now we can put all those components to work and introduce task-based redundancy resolution.

3.1 TBRR Concept

The concept of TBRR revolves around utilizing redundancy to make the robot execute the task successfully. From the user's point of view, his role is greatly simplified to specifying the *critical* values for the constraints and the *desired* values for the task requirements. Then, TBRR attempts to allocate the extra resources to help the robot perform the task without violating any constraint and satisfying the speed, accuracy, and force requirements. The remaining resources can then be utilized to minimize the energy consumption or other criteria as desired. Figure 1 illustrates the concept of TBRR.

The red (or shaded) regions correspond to violations of the system constraints, shown here are joint limit, obstacle, and singularity. The yellow (or diagonal patterned) regions denote violations of the task requirements. Therefore only the null space in the middle is deemed acceptable in terms of satisfying the constraints and task requirements. The best configuration according to the efficiency-related criterion is then selected from this acceptable null space.

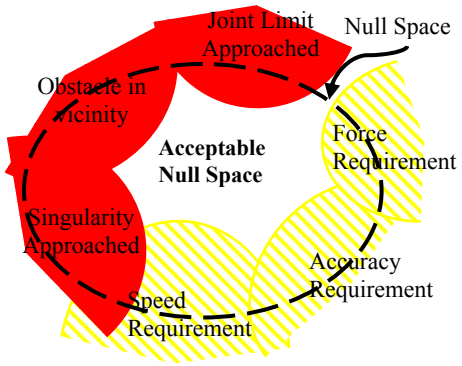

Figure 1. TBRR Concept of Constraints and Task Requirements

3.2 TBRR Scheme

In earlier studies, we have proposed a performance-based multi-criteria RRT [11]. To find an optimal solution, this RRT uses a two-step approach consisting of a generation of candidate inverse kinematics solutions and a selection of solution via evaluations of performance criteria. Basically, this method generates a finite number of solutions in the null space called "options" by means of direct search [9], ranks these options based on the performance objective function, and then selects the best ranked option. TBRR borrows this two-step approach. However, the solution selection step instead evaluates the system constraints and task requirements and filters out non-compliant options. This results in an acceptable null space that is smaller than the original null space. This two-step TBRR scheme is depicted in Figure 2 and the detailed discussion is followed.

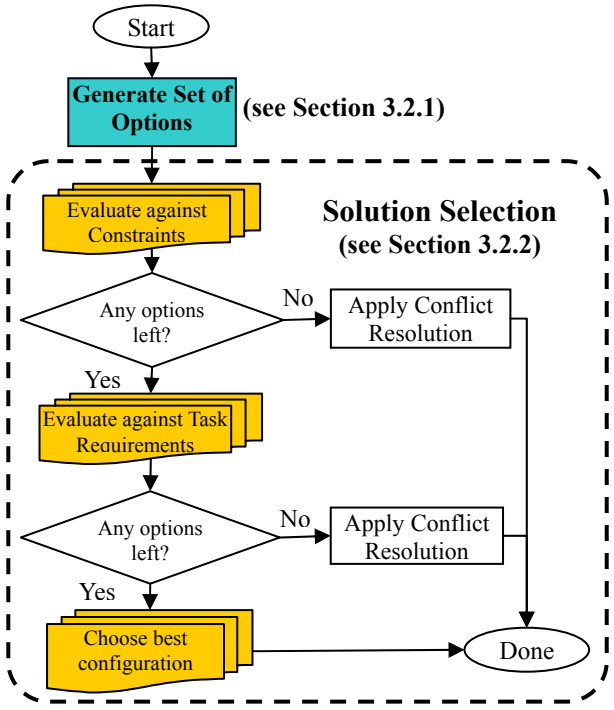

Figure 2. Flow Chart Of TBRR Implementation.

3.2.1 Generate Options

Similar to the generate options scheme implemented earlier in [11], a two-level scheme is also employed in the generation of the candidate inverse kinematics solutions. The first level uses the pseudoinverse of the Jacobian in solving the position-level inverse kinematics.

Let $\bar{\theta}$ denote the current set of joint variables so that

$$f(\bar{\theta}) = \bar{x} \tag{20}$$

gives the current EEF position. The difference between the actual and the desired EEF positions is given by

$$\delta x = x - \bar{x}\,. \tag{21}$$

The required changes in the joint variables can be found using

$$\delta\theta = J^{\#}\delta x \tag{22}$$

where $J^{\#}$ is the pseudoinverse of the Jacobian. The new set of joint variables thus becomes

$$\theta = \bar{\theta} + \delta\theta \tag{23}$$

which is iterated through the forward kinematics, using (20)-(23), until

$$\left\|x - \bar{x}\right\| < \varepsilon \tag{24}$$

where ε is a small positive-valued error tolerance for the EEF constraints.

The second level uses the "direct search" technique [9] which is accomplished by introducing small perturbations to joint variables in a systematic fashion. These joint perturbations result in a change and an error in the EEF position. The required changes in the joint variables to compensate for this change can be computed by (22). For different sets of joint perturbations, we end up with a finite set of options that satisfy the same EEF constraints. The Generate Options scheme creates a finite number of inverse kinematics solutions around the current configuration of the manipulator without moving its end-effector. This is called *self-motion* and is shown in Figure 3.

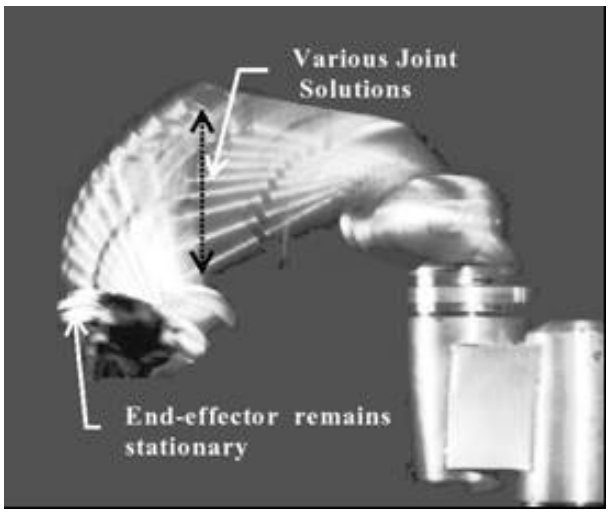

Figure 3. A Redundant 7-DOF Arm Performing Self-Motion.

3.2.2 Solution Selection

The next step is to select the best solution from the set of generated options. Traditionally, the best solution is determined by evaluating a composite performance index. Let $\psi_i(\theta): \mathcal{R}^n \to \mathcal{R},\ \ i = 1\ldots L$ denote the i^{th} performance criterion defined as a function of the configuration $\theta \in \mathcal{R}^n$. For the sake of simplicity and without loss of generality, we assume that all performance criteria are to be minimized. Let $\theta^1, \theta^2, \ldots \theta^K$ be the set of generated options from the previous section. A composite index for the j^{th} configuration (option) can then be defined as

$$G_j = \sum_{i=1}^{L} w_i \psi_i\left(\theta^j\right), \quad j = 1\ldots K \tag{25}$$

where $0 \le w_i \le 1$ is the relative weight of the i^{th} criterion. The weights can be fixed throughout the entire operation or dynamically changing according to some algorithms (see [3] for example). Another type of composite index is presented in [11] and is repeated here.

$$H_j = \frac{1}{L}\sum_{i=1}^{L} w_i \bar{\psi}_i\left(\theta^j\right), \quad j = 1\ldots K \tag{26}$$

where

$$\bar{\psi}_i\left(\theta^j\right) = \frac{\psi_i\left(\theta^j\right) - \min_j\left\{\psi_i\left(\theta^j\right)\right\}}{\max_j\left\{\psi_i\left(\theta^j\right)\right\} - \min_j\left\{\psi_i\left(\theta^j\right)\right\}}\,. \tag{27}$$

Notice that $\bar{\psi}_i\left(\theta^j\right)$ is the normalized form of $\psi_i\left(\theta^j\right)$ and $0 \le H_j \le 1$. The best configuration is the one that yields the minimum value for G_j or H_j. It should be noted that this two-level approach is a sub-optimal but efficient way of resolving redundancy without the need for calculating the gradients of the objective functions.

In TBRR, however, we apply the system constraints and task requirements to the redundancy resolution scheme. For each option, the constraint-based criteria are evaluated against the critical values specified by the user. The options that do not satisfy all the constraints are filtered out and only the options that do are passed on to the next step. Next, the robot capabilities are estimated and then evaluated against the desired values of the task requirements specified by the user. Again, the options that do not satisfy all the task requirements are filtered out. The best configuration can then be chosen from the remaining options by using the efficiency criterion to achieve long-term efficient operation or any other criterion as desired.

It is not unusual that none of the generated options satisfies all constraints and/or task requirements. In that case, two alternatives exist. The first one is to notify the Generate Options scheme to generate a new set of options and apply TBRR again until at least one option that satisfies all constraints and task requirements is found. There is, however, no guarantee that one such option exists. It is quite possible that in that neighborhood the robot cannot physically satisfy all the constraints and task requirements. This brings us to the second alternative: conflict

resolution. In conflict resolution, we would choose the option that *least violates* the constraints and/or task requirements.

The concept of conflict resolution of constraints and task requirements is identical. Therefore, we will only show that for the constraints and the same will be applied for the task requirements as well. For simplicity and without loss of generality, we assume that all constraint-based criteria are to be maximized. For each constraint, a *deviation* from the critical value is defined as:

$$z_i = \begin{cases} 0; & a_i \geq c_i \\ \left(\dfrac{c_i - a_i}{c_i} \right) \times 100\%; & a_i < c_i, c_i > 0 \\ \left(c_i - a_i \right); & a_i < c_i, c_i = 0 \end{cases} \quad (28)$$

where a_i is the actual value, c_i is the critical value of constraint i. Here if the actual value is less than the critical value (i.e. the constraint is already satisfied), then the deviation is zero. If not, then the deviation will have some positive value. Then, all the deviations can be added in a linear fashion to form the overall score.

$$Z = \sum_i w_i z_i \quad (29)$$

where w_i is the weight for constraint i. From (28) and (29), if the score is 0, then no constraint is violated. Larger scores mean higher degrees of constraint violation. Therefore, the solution with the lowest score will be chosen.

Normally, all the weights should be set to 1.0 because all the constraints or task requirements (at their respective levels) are considered equally important. However, one could argue that obstacle avoidance is more critical than singularity avoidance since if obstacles are not avoided, it could result in physical damages. In that case, the weight of the constraint associated with obstacle avoidance can be raised higher.

3.2.3 Buffering

Most RRTs employ local optimization that relies only on the information at the time instant. TBRR is no exception. TBRR will attempt to satisfy whatever task requirement values that have been passed on to the scheme at that instant. Basically, TBRR does not influence the solution until one or more task requirement values approach the associated robot capabilities. This however can cause a delay and lead to unsatisfactory solutions because the robot may already be in a neighborhood where it cannot physically satisfy those specific task requirements.

Therefore, TBRR would in general be more effective if it is supplied with the task requirement values plus some *buffers*. This is like giving the manipulator a *performance reserve* or an advance warning for each task requirement. For example if the force requirement is 200 N, then it may be better to add a buffer of say 20 N and pass the force requirement value of 220 N onto TBRR. This will allow the robot to adjust its configuration when its estimated force capability declines to 220 N, instead of 200 N. The effect of buffering will be demonstrated in the simulations.

Some may dispute that these buffer values are subjective and their inclusions introduce another burden to the user of the scheme. It is argued here, however, that because these buffers have physical meanings that the user can easily relate to, buffering is much more straightforward than assigning the weights to performance criteria. In addition, a development of an *adaptive buffering* scheme is underway. Adaptive buffering would alleviate this burden from the user.

3.3 Software Implementation

TBRR software was developed using OSCAR (Operational Software Components for Advanced Robotics), which is an object-oriented framework implemented in C++ language for developing control software for intelligent machines [10][15]. This framework presents the applications developer with a generalized and extensible environment that supports mathematical abstractions, generalized forward and inverse kinematics, generalized dynamics, performance criteria, etc. for serial manipulators. TBRR software implementation abides by OSCAR's principle concepts of generality, extensibility, and modularity. As a result, TBRR can be applied to serial manipulators of any geometry. Additional constraints or task requirements can also be easily added as desired.

4. SIMULATIONS

In this section, we evaluate the performance of the proposed redundancy resolution scheme by computer simulations. The goal of these simulations is to demonstrate that TBRR can improve the system performance over traditional RRTs when it comes to satisfying system constraints and task requirements.

In order to accurately gauge the effectiveness of TBRR, we will run simulations on a spatial 10-DOF robot using traditional RRTs and TBRR and compare the results. This 10-DOF robot was chosen due to its complexity. The RRTs chosen in this comparative study are the pseudoinverse (PI), an RRT using the composite performance index in (26) with a fixed weighting (FW) scheme. The pseudoinverse solution is used for baseline comparison. FW uses the same "generate options" scheme as TBRR and thus the only difference between TBRR and FW lies in how they select the best solution. By doing this, we emphasize the effect of TBRR and not the intricacy of the underlying search algorithm.

The robot's EEF is to follow the circular path with the diameter of 1 m, which is the lid of the barrel shown in Figure 4 without changing its orientation. The position and size of the barrel are chosen so that its far end is close to the boundary of the robot workspace, thus forcing the robot to approach its boundary singularities. The path is trapezoidal at the velocity level and is divided into a total of 600 step points. The total execution time is 12 seconds.

For FW, we use the JRA and MOM criteria to avoid joint limits and singularities. In addition, the following two performance criteria may be included in hope of improving the robot's EEF force capability and accuracy.

Torque Efficiency (TEF)

$$\gamma_{TEF} = u^T \left(JJ^T \right) u \quad (30)$$

where u is the unit vector in the direction of interest. TEF is the reciprocal of the force transmission ratio criterion defined in

[5]. Minimizing TEF will lead to increased force capability in u direction. In our simulations, u is set to be in the Z direction.

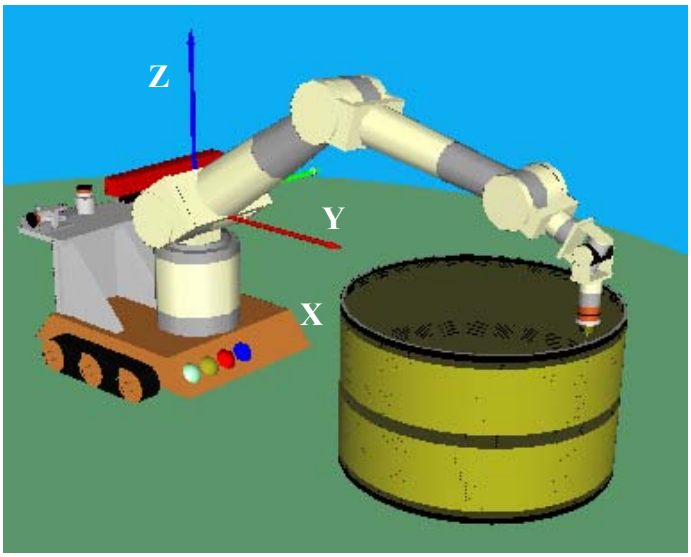

Figure 4. 10-DOF robot tracking a circular path.

Generalized Load Stiffness (GLS)

$$\gamma_{GLS} = \left(\sum_{i=1}^{m} \lambda_i^2 \right)^{-\frac{1}{2}} \tag{31}$$

where λ_i the i^{th} eigenvalue of the stiffness matrix [17]. Minimizing GLS usually leads to smaller EEF deflections due to joint compliances; thus increased accuracy.

In the experiments to be followed, we ran FW with different set of weights. Let w_1, w_2, w_3, and w_4 be the weights of JRA, MOM, GLS, and TEF criteria used in FW, respectively. These sets of weights are listed in Table I.

TABLE I. SETS OF WEIGHTS FOR FW

	w_1	w_1	w_3	w_4
FW1	1	1	0	0
FW 2	1	1	1	0
FW 3	1	1	0	1
FW 4	1	1	1	1

4.1 Experiment I

The task requirements in terms of speed, force, and accuracy are listed in Table II.

TABLE II. TASK REQUIREMENTS FOR EXPERIMENT I.

	Max. Speed (m/s; deg/s)	Force (N; N-m)	Accuracy (mm; deg)
X	0.4	0	1.5
Y	0.4	0	1.5
Z	0	250	1.5
Roll	0	0	2
Pitch	0	0	2
Yaw	0	0	2

The simulation results are summarized in Table III. The numbers in the table represent the number of step points (out of 600) in which a certain task requirement is not satisfied. Other task requirements not listed in the table are satisfied at all step points and therefore need not be compared. The *total* (in the fifth column) is merely a sum of the numbers in the previous three columns. It represents the total number of task requirement violations for a given method along the trajectory. If all task requirements are considered equally important, then the total can be used as a single performance metric, indicating how well each method performs (the lower the total, the better). One exception is when any constraint violation occurs. Since constraints have higher priority, a method that produces constraint violation will be deemed a failure and inferior to a method without constraint violation regardless of the total. The asterisks next to PI and FW4 indicate that the joint limit constraint was violated during the operation and thus were physically impossible if we were to run them on a real robot. In TBRR1, buffering is not used, i.e. the task requirement values in Table I are used directly in the TBRR scheme. TBRR2 here adds a buffer of 10 N to the force requirement in Z.

TABLE III. NUMBER OF POINTS WHERE TASK REQUIREMENTS ARE NOT MET FOR EXPERIMENT I.

Method	Velocity in X	Force in Z	Accuracy in Z	Total
*PI**	*0*	*465*	*158*	***623***
FW1	0	174	0	**174**
FW2	0	316	0	**316**
FW3	67	153	0	**220**
*FW4**	*0*	*87*	*0*	***87***
TBRR1	0	13	0	**22**
TBRR2	0	0	0	**0**

4.2 Experiment II

To demonstrate the flexibility of TBRR, we changed some of the task requirements to the values shown in Table IV. Note that we reduce the force requirement in the Z direction and increase the accuracy requirements (by reducing the acceptable errors) in the X, Y, and Z directions from Experiment I.

TABLE IV. TASK REQUIREMENTS FOR EXPERIMENT II.

	Max. Speed (m/s; deg/s)	Force (N; N-m)	Accuracy (mm; deg)
X	0.4	0	1.2
Y	0.4	0	1.2
Z	0	100	1.2
Roll	0	0	2
Pitch	0	0	2
Yaw	0	0	2

The results for Experiment II are shown in Table V. TBRR1 again indicates TBRR without buffering. TBRR2 adds a buffer of 0.1 mm for the accuracy requirements in the Y and Z directions.

TABLE V. NUMBER OF POINTS WHERE TASK REQUIREMENTS ARE NOT MET FOR EXPERIMENT II.

Method	Velocity in X	Accuracy in Y	Accuracy in Z	Total
*PI**	*0*	*101*	*139*	***340***
FW1	0	105	132	**237**
FW2	0	168	211	**379**
FW3	67	65	81	**213**
*FW4**	*0*	*163*	*214*	***377***
TBRR1	0	68	62	**130**
TBRR2	0	51	45	**96**

As seen from both experiments, TBRR yielded much superior results to traditional RRTs that use the weighted sum method as a means to cope with multiple criteria. With appropriately chosen values of buffers, TBRR performed even better. In Experiment I, with TBRR and buffering, the robot satisfies the task requirements at every step point. FW4 would have been the closest competition to TBRR had it not violated the joint limit constraint. As a result, FW1 was the second best behind TBRR. With changes in task requirements in Experiment II, TBRR with buffering still performs the best even though it cannot satisfy the task requirements at all step points. It is interesting to note that FW3 came in second behind TBRR in Experiment II while FW1 was second in Experiment I. One cannot predict which combination of weights will yield the best results when there are changes in the task requirements and this is a major drawback of traditional RRTs.

We can clearly see the advantages of TBRR over traditional RRTs. Not only does it better satisfy the task requirements, but it also is able to adapt more effectively. For traditional RRTs, in order to best satisfy the task requirements, the user needs to figure a new combination of performance criteria when the task requirements change (as seen from the experiments that FW1 was best for Experiment I whereas FW3 was best for Experiment II). This process is not trivial and requires in-depth experience. With TBRR, when the task requirements change, the user just needs to supply the TBRR scheme with a new set of task requirements and optionally a set of buffers. In real-world complex tasks, since the task requirements can change rapidly, TBRR can save a lot of deployment time and money.

In both experiments, an operational frequency of approximately 50 Hz was achieved by the TBRR scheme on a PC with Athlon XP 2400 processor with 512 MB of RAM running Windows XP. By contrast, FW with all four performance criteria ran at around 95 Hz on the same machine. Although TBRR does not run as fast as FW, its computational speed is acceptable for supervisory control applications.

5. CONCLUSION

We have presented the concepts of system constraints, task requirements, and task executability and applied them in a new task-based decision making scheme for redundant manipulators, called Task-Based Redundancy Resolution (TBRR). TBRR uses system constraints and task requirements to guide the redundancy resolution process. It computes the robot capabilities and compares them with the task requirements to ensure the task can be executed. TBRR has been shown with computer simulations of a 10-DOF robot to be more effective and more responsive than traditional RRTs using the weighted sum method.

Currently, two areas of further research are being actively pursued. One is the development of new robot capability estimations using a method called *vector expansion*, which is based on the ellipsoid expansion method [1]. The vector expansion method is aimed at providing more accurate estimates than the ellipsoid method used in this paper. Yet it is computationally efficient enough for use in real-time monitoring and control, and thus in TBRR. The other effort tries to improve the performance of TBRR through a method called *adaptive buffering*. Adaptive buffering uses soft computing techniques (fuzzy logic, evolutionary algorithms, etc.) and online information to actively adjust the sizes of the buffers in real-time.

ACKNOWLEDGMENTS

This work was supported by funding from Department of Energy (grant no. DE-FG04-94EW37966) and Texas Higher Education Coordinating Board (grant no. ATP 003658-0034-2001).

REFERENCES

[1] Bowling, A. and Khatib, O., "Analysis of the Acceleration Characteristics of Non-Redundant Manipulators," *Proc. of IEEE/RSJ IROS Conf.*, pp. 323-328, 1995.

[2] Cheng, F.-T., Sheu, R.-J., Chen, T.-H., and Kung, F.-C., "The improved compact QP method for resolving manipulator redundancy," *Proc. of IEEE/RSJ/GI IROS Conf.*, pp. 1368-1375, 1994.

[3] Cheng, F.-T., Shih, M.-S., Kung, F.-C., and Sun, Y.-Y., "The improved parallel scheme for multiple-goal priority considerations of redundant manipulators," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 2409-2414, 1997.

[4] Chiacchio, P. and Concilio, M., "The Dynamic Manipulability Ellipsoid for Redundant Manipulators," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, pp. 95-100, 1998.

[5] Chiu, S., "Kinematic characterization of manipulators: an approach to defining optimality," *Proc. of IEEE Int. Conf. on Robotics and Automation*, Philadelphia, PA, pp. 828-833, 1988.

[6] Hanson, M.L. and Tolson, R.H., "Using a fuzzy supervisor to optimize multiple criteria in redundant robots," *Proc. of the 10th IEEE Int. Symp. On Intelligent Control*, pp. 145-150, 1995.

[7] Harden, T., Kapoor, C., and Tesar, D., "Experimental evaluation of a criteria-based obstacle avoidance scheme," *Proc. of ASME Design Engineering Technical Conferences*, Las Vegas, NV, 1999.

[8] Hollerbach, J.M. and Suh, K.C., "Redundancy resolution of manipulators through torque optimization," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 1016-1021, 1985.

[9] Hooper, R. and Tesar, D., *Multicriteria Inverse Kinematics for General Serial Robots,* Ph.D. Dissertation, University of Texas at Austin, 1994.

[10] Kapoor, C. and Tesar, D., *A reusable operational software architecture for advanced robotics*, Ph.D. Dissertation, University of Texas at Austin, 1996.

[11] Kapoor, C., Cetin, M., and Tesar, D., "Performance based redundancy resolution with multiple criteria," *Proc. of ASME Design Engineering Technical Conference*, Atlanta, Georgia, 1998.

[12] Liegeois, A., "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics*, v. SMC-7, n.12, pp. 868-871, 1977.

[13] McGhee, S., Chan, T.F., Dubey, R.V., and Kress, R.L., "Probability-based weighting of performance criteria for a redundant manipulator," *Proc. of IEEE Int. Conf. on Robotics and Automation*, v. 3, pp. 1887-1894, 1994.

[14] Nakamura, Y., Hanafusa, H., and Yoshikawa, T., "Task priority based redundancy control of robot manipulators," *Int. Journal of Robotics Research*, v. 6, n. 2, pp. 3-15, 1987.

[15] OSCAR version 2.0 online tutorial and documentation: http://www.robotics.utexas.edu/rrg/downloads/software/oscarv.2/

[16] Pamanes J.A., Zeghloul, S., "Optimal placement of robotic manipulators using multiple kinematic criteria," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 933-938, 1991.

[17] Pryor, M., Van Doren, M., and Tesar, D., "Manipulator performance criteria based on kinematic, dynamic, and compliance models," *Proc. of ASME Design Engineering Technical Conference*, Las Vegas, NV, 1999.

[18] Siciliano, B. and Slotine, J.-J. E., "A general framework for managing multiple tasks in highly redundant robotic systems," *Proc. of Int. Conf. on Advanced Robotics*, pp. 1211-1216, 1991.

[19] Whitney, D.E., "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on Man-Machine Systems*, v. MMS-10, n. 2, pp. 47-53, 1969.

[20] Yoshikawa, T., "Manipulability and redundancy control of robotic mechanisms," *Proc. of IEEE Int. Conf. on Robotics and Automation,* pp. 1004-1009, 1985.