Planning Motions with Intentions

Yoshihito Koga†, Koichi Kondo‡, James Kuffner† and Jean-Claude Latombe†
† Robotics Laboratory Department of Computer Science, Stanford University
Stanford, CA 94305, USA
‡ R & D Center, Toshiba Corporation, 4-1 Ukishima-cho, Kawasaki, 210, Japan

Abstract

We apply manipulation planning to computer animation. A new path planner is presented that automatically computes the collision-free trajectories for several cooperating arms to manipulate a movable object between two configurations. This implemented planner is capable of dealing with complicated tasks where regrasping is involved. In addition, we present a new inverse kinematics algorithm for the human arms. This algorithm is utilized by the planner for the generation of *realistic* human arm motions as they manipulate objects. We view our system as a tool for facilitating the production of animation.

1 Introduction

Mundane details keep us from vigorously attacking bigger ideas. This is the motivation for achieving task-level animation. From task-level descriptions, the animation of figures in a scene can be automatically computed by an appropriate motion planner. The animator can thereby concentrate on creating imaginative graphics, rather than labouring over the chore of moving these figures in a realistic and collision-free manner. In this paper we present our efforts towards realizing a subset of this ultimate goal - the automatic generation of human and robot arm motions to complete manipulation tasks.

Why study manipulation with arms? Human figures often play an integral role in computer animation. Consequently, there are arm motions and more specifically manipulation motions to animate. Another major application is in ergonomics. Since most products are utilized, assembled, maintained, and repaired by humans, and require for most cases some action by the human arms, by simulating and viewing these arm motions through computer graphics, one can evaluate the design of the product in terms of its usability. This will reduce the number of mock-up models needed to come up with the final design. Again, this would allow the designer more time towards creating high-quality products.

Unlike the motion of *passive* systems like falling objects or bouncing balls, the motion of human and robot arms for the purpose of manipulation are "motions with intentions". The arms move not through some predictable trajectory due to the laws of physics (as is the case with a falling object [2, 9]) but with the *intention* of completing some task. A planner is needed to determine how the arms must move to complete the task at hand. Although there has been previous work on simulating walking and lifting motions, this is the first attempt to automatically generate complex manipulation motions.

Our problem is thus to find a collision-free path for the arms to grasp and then carry some specified movable object from its initial location to a desired goal location. This problem is known as the multi-arm manipulation planning problem [12, 7]. A crucial difference, relative to more classical path planning, is that we must *account for the ability of the arms to change their grasp of the object.* Indeed, for some tasks the arms may need to ungrasp the object and regrasp it in a new manner to successfully complete the motion.

We present a new planner that solves this multi-arm manipulation problem [12]. The planner needs as input the geometry of the environment, the initial and goal configurations of the movable object and arms, a set of potential grasps of the movable object, and the *inverse kinematics of the arms.* With appropriate book-keeping, the animator would simply specify the goal configuration of the movable object (a task-level description) to generate the desired animation.

The planning approach is flexible in regards to the arm types that can be considered. The only restriction is that the arms must have an inverse kinematics algorithm. Thus, in addition to the planner, we present a new inverse kinematics algorithm for the human arms based on results from neurophysiology. This algorithm resolves the redundancy of the human arms by utilizing a sensorimotor transformation model [29, 30]. The result is the automatic animation of human arm manipulation tasks.

In addition to the motion planning aspect, we address the issue of producing natural motions when human figures are animated. By applying results from neurophysiology to various parts of the planner, for instance the inverse kinematics, we hope to achieve a good approximation of naturalness. We believe that the geometry of the motions produced by the planner is in fact quite natural. [1]

Fig. 1 is a series of snapshots along a manipulation path computed by our planner. Once the necessary models of the environment are read by the planner, the input from the animator is simply the goal location for the eye glasses, in this case getting placed on the head. Note, the planner found automatically that the arms must ungrasp and regrasp the glasses in order to complete the task.

Section 2 relates our efforts to previous work. Section 3 describes the details of our motion planner. Section 4 is the derivation of the human arm inverse kinematics. Section 5 is a discussion of how our system takes natural human arm motion into consideration. Finally, Section 6 presents results obtained with the planner.

2 Related Work

Planning *motions with intentions* for robot and human arm manipulation is related to several different areas of research. We roughly classify this related work into three categories, manipulation planning, animation of human figures, and neurophysiology.

Manipulation Planning: The use of path planning to automatically generate graphic animation was already suggested in [18]. Research strictly addressing manipulation planning is fairly recent. The first paper to tackle this problem is by Wilfong [34]. It considers a single-body robot translating in a 2D workspace with multiple movable objects. The robot, the movable objects and obstacles are modelled as convex polygons. In order for the movable objects to reach their specified goal the robot must "grasp" and carry them there. Wilfong shows that planning a manipulation path to bring the movable objects to their specified goal locations is PSPACE-hard. When there is a single movable object, he proposes a complete algorithm that runs in $O(n^3 \log^2 n)$ time, where n is the total number of vertices of all the objects in the environment. Laumond and Alami [15] propose an $O(n^4)$ algorithm to solve a similar problem where the robot and the movable object are both discs and the obstacles are polygonal.

Our work differs from this previous research in several ways. Rather than dealing with a single robot, we consider the case of multiple human and robot arms manipulating objects in a 3D workspace. In addition, whereas the previous work is more theoretical in nature, our focus is more on developing an effective approach to solving manipulation tasks of a complexity comparable to those encountered in everyday situations (e.g. picking and placing objects on a table).

Regrasping is a vital component in manipulation tasks. Tournassoud, Lozano-Pérez, and Mazer [32] specifically address this problem. They describe a method for planning a sequence of regrasp operations by a single arm to change an initial grasp into a goal grasp. At every regrasp, the object is temporarily placed on a horizontal table in a stable position selected by the planner. We too need to plan regrasp operations. However, the only regrasping motions we consider avoid contact between the object and the environment; they necessarily involve multiple arms (e.g. both human arms).

Grasp planning is potentially an important component of manipulation planning. In our planner, grasps are only selected from a finite predefined set. An improvement for the future will be to include the automatic computation of grasps. A quite substantial amount of research has been done on this topic. See [22] for a commented list of bibliographical references.

Animation of Human Figures: Human gaits have been successfully simulated. For example, Bruderlin and Calvert [5] have proposed a hybrid approach to the animation of human locomotion which combines goal-directed and dynamic motion control. McKenna and Zeltzer [21] have successfully simulated the gait of a virtual insect by combining forward dynamic simulation and a biologically-based motion coordination mechanism. Control algorithms have been successfully applied to the animation of dynamic legged locomotion [25]. While dynamic models and the use of motor coordination models have been successfully applied to a wide range of walking motions, such a strategy has yet to be discovered to encompass human arm motions.

For simulating the motion of human arms, there exist methods for specific tasks. For example, Lee et al. [17] have focused on the simulation of arm motions for lifting based on human muscle models. Their method considers such factors as comfort level, perceived exertion, and strength. Our approach is to sim-

[1] This paper does not consider the velocity distribution along the planned motions.

ulate natural arm motions from the point of view of kinematics, that is we make no consideration of dynamics and muscle models. We justify this approach in Section 5.

There has been previous work in applying motion planning algorithms to animating human figures. Ching and Badler [6] present a motion planning algorithm for anthropometric figures with many degrees of freedom. Essentially, they use a sequential search strategy to find a collision-free motion of the figure to a specified goal configuration. They do not consider manipulation or imposing naturalness on the motions.

The AnimNL project at the University of Pennsylvania [33] is working to automate the generation of human figure movements from natural language instructions. Their focus is mainly on determining the sequence of primitive actions from a high-level description of the task. They utilize models to create realistic motions, however they do not consider complex manipulation motions.

Neurophysiology: There are many scientists in psychology and neurophysiology working to determine how the human brain manages to coordinate the motion of its limbs. Soechting [31] gives a good survey of various empirical studies and their results for human arm motions.

One relevant finding is the sensorimotor transformation model devised by Soechting and Flanders [29, 30]. They found that the desired position of the hand roughly determines the arm posture. Our inverse kinematics algorithm for the human arm is based on this result.

3 Manipulation Planner

In this section we present our manipulation planning algorithm. The method applies to any system of arms as long as they have an inverse kinematics solution.

3.1 Problem Statement

We now give a rather formal formulation of the multi-arm manipulation planning problem. We consider only a single movable object, but for the rest, our presentation is general.

The environment is a 3D workspace $\mathcal{W}$ with p arms $\mathcal{A}_i$ $(i = 1, \cdots, p)$, a single movable object $\mathcal{M}$, and q static obstacles $\mathcal{B}_j$ $(j = 1, \cdots, q)$. The object $\mathcal{M}$ can only move by having one or several of the arms grasp and carry it to some destination.

Let $\mathcal{C}_i$ and $\mathcal{C}_{obj}$ be the C-spaces (configuration spaces) of the arms $\mathcal{A}_i$ and the object $\mathcal{M}$, respectively [19, 16]. Each $\mathcal{C}_i$ has dimension n_i, where n_i is the number of degrees of freedom of the arm $\mathcal{A}_i$, and $\mathcal{C}_{obj}$ has dimension 6. The *composite C-space* of the whole system is $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_p \times \mathcal{C}_{obj}$. A configuration in $\mathcal{C}$, called a *system configuration*, is of the form $(\boldsymbol{q}_1, \cdots, \boldsymbol{q}_p, \boldsymbol{q}_{obj})$, with $\boldsymbol{q}_i \in \mathcal{C}_i$ and $\boldsymbol{q}_{obj} \in \mathcal{C}_{obj}$.

We define the *C-obstacle region* $\mathcal{CB} \subset \mathcal{C}$ as the set of all system configurations where two or more bodies in $\{\mathcal{A}_1, \cdots, \mathcal{A}_p, \mathcal{M}, \mathcal{B}_1, \cdots, \mathcal{B}_q\}$ intersect.[2] We describe all bodies as closed subsets of $\mathcal{W}$; hence, $\mathcal{CB}$ is a closed subset of $\mathcal{C}$. The open subset $\mathcal{C} \setminus \mathcal{CB}$ is denoted by $\mathcal{C}_{free}$ and its closure by $cl(\mathcal{C}_{free})$.

For the most part we require that the arms, object, and obstacles do not contact one another. However, $\mathcal{M}$ may touch stationary arms and obstacles for the purpose of achieving static stability. $\mathcal{M}$ may also touch arms when it is being moved. This is to achieve grasp stability and $\mathcal{M}$ can only make contact with the end-effector of each grasping arm (grasping may involve one or several arms). No other contacts are allowed.

This leads us to define two subsets of $cl(\mathcal{C}_{free})$:
- The *stable space* $\mathcal{C}_{stable}$ is the set of all legal configurations in $cl(\mathcal{C}_{free})$ where $\mathcal{M}$ is statically stable. $\mathcal{M}$'s stability may be achieved by contacts between $\mathcal{M}$ and the arms and/or the obstacles.
- The *grasp space* $\mathcal{C}_{grasp}$ is the set of all legal configurations in $cl(\mathcal{C}_{free})$ where one or several arms rigidly grasp $\mathcal{M}$ in such a way that they have sufficient torque to move $\mathcal{M}$. $\mathcal{C}_{grasp} \subset \mathcal{C}_{stable}$.

There are two types of paths, transit and transfer paths, which are of interest in multi-arm manipulation:
- A *transit path* defines arm motions that do not move $\mathcal{M}$. Along such a path $\mathcal{M}$'s static stability must be achieved by contacts with obstacles and/or stationary arms. Examples of such a path involve moving an arm to a configuration where it can grasp $\mathcal{M}$ or moving an arm to change its grasp of $\mathcal{M}$. A transit path lies in the cross-section of $\mathcal{C}_{stable}$ defined by the current fixed configuration of $\mathcal{M}$.
- A *transfer path* defines arm motions that move $\mathcal{M}$. It lies in the cross-section of $\mathcal{C}_{grasp}$ defined by the attachment of $\mathcal{M}$ to the last links of the grasping arms. During a transfer path, not all moving arms need grasp $\mathcal{M}$; some non-grasping arms may be moving to allow the grasping arms to move without collision.

[2] We regard joint limits in $\mathcal{A}_i$ as obstacles that only interfere with the arms' motions.

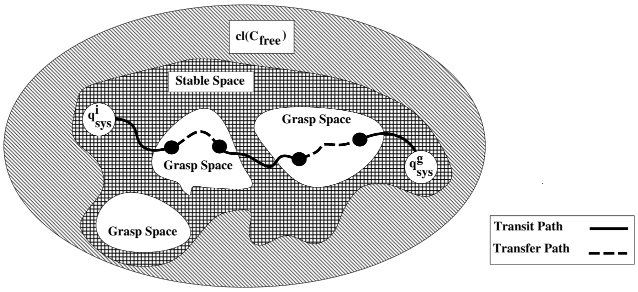

Figure 2. Components of a manipulation path and their relation to the subspaces of $cl(\mathcal{C}_{free})$.

A *manipulation path* is an alternate sequence of transit and transfer paths that connects an initial system configuration, $\boldsymbol{q}^i_{sys}$, to a goal system configuration, $\boldsymbol{q}^g_{sys}$ (see Fig. 2). Some paths in this sequence may be executed concurrently as long as this does not yield collisions.

In a multi-arm manipulation planning problem, the geometry of the arms, movable object, and obstacles is given, along with the location of the obstacles. The goal is to compute a manipulation path between two input system configurations.

3.2 Planning Approach

We now describe our approach for solving the multi-arm manipulation planning problem. This approach embeds several simplifications. As a result the corresponding planner is not fully general. Throughout our presentation, we carefully state the simplifications that we make. Some of them illustrate the deep intricacies of multi-arm manipulation planning.

Overview: A manipulation path alternates transit and transfer paths. Each path may be seen as the plan for a subtask of the total manipulation task. This yields the following two-stage planning approach: first, generate a series of subtasks to achieve the system goal configuration; second, plan a transit or transfer path for each subtask. An informal example of a series of subtasks is: grab $\mathcal{M}$, carry it to an intermediate location, change grasp, carry $\mathcal{M}$ to its goal location, ungrasp.

Unfortunately, planning a series of subtasks that can later be completed into legal paths is a difficult problem. How can one determine whether a subtask can be completed without actually completing it? We settle for a compromise. Our approach focuses on planning a sequence of transfer tasks that are guaranteed to be completed into transfer paths. In fact, in the process of identifying these tasks, the planner also generates the corresponding transfer paths. With the transfer tasks specified, the transit tasks are immediately defined: they link the transfer paths together along with the initial and goal system configuration. It only remains to compute the corresponding transit paths.

The assumption underlying this approach is that there exists a legal transit path for every transit task. Actually, in a 3D workspace, this is often the case. If the assumption is not verified, the planner may try to generate another series of transfer tasks, but in our current implementation it simply returns failure.

Restrictions on grasps: To simplify the selection of transfer tasks, we impose a restriction on grasps. The various possible grasps of $\mathcal{M}$ are given as a finite *grasp set*. Each grasp in this set describes a rigid attachment of the end-effector(s) of one or several arms with $\mathcal{M}$. For example, if $\mathcal{M}$ is considered too heavy or too bulky to be moved by a single arm, each grasp in the grasp set may indicate the need for a two-arm grasp. If the end-effector consists of multi-joint fingers, their joint values to achieve the particular grasp are specified.

Generating transfer tasks: The generation of the transfer tasks is done by planning a path τ_{obj} of $\mathcal{M}$ from its initial to its goal configuration. During the computation of τ_{obj}, all the possible ways of grasping $\mathcal{M}$ are enumerated and the configurations of $\mathcal{M}$ requiring a regrasp are identified.

The planner computes the path τ_{obj} so that $\mathcal{M}$ avoids collision with the static obstacles $\mathcal{B}_j$. This is done using RPP (Randomized Path Planner), which is thus a component of our planner. RPP is described in detail in [3, 16].

RPP generates τ_{obj} as a list of adjacent configurations in a fine grid placed over $\mathcal{C}_{obj}$ (the 6D C-space of $\mathcal{M}$), by inserting one configuration after the other starting with the initial configuration of $\mathcal{M}$. The original RPP only checks that each inserted configuration is collision-free. To ensure that there exists a sequence of transfer paths moving $\mathcal{M}$ along τ_{obj}, we have modified RPP. The modified RPP also verifies that at each inserted configuration, $\mathcal{M}$ can be grasped using a grasp from the input grasp set. This is done in the following way. A *grasp assignment* at some configuration of $\mathcal{M}$ is a pair associating an element of the grasp set defined for $\mathcal{M}$ and the identity of the grasping arm(s). Note that the same element of the grasp set may yield different grasp assignments involving different arms. The planner enumerates all the grasp assignments at the initial configuration of $\mathcal{M}$ and keeps a list of those which can be achieved without collision between the grasping arm(s) and the obstacles, and between the grasping arms should there be more than one. We momentarily ignore the possibility that the grasping arm(s) may collide with the other non-grasping arms. The list of possible grasp assignments is associated with the initial configuration. Prior to inserting any new configuration in the path being generated, RPP prunes the list of grasp assignments attached to the previous configuration by removing all those which are no longer possible at the new configuration. The remaining sublist, if not empty, is associated with this configuration which is appended to the current path.

If during a down motion of RPP (a motion along the negated gradient of the potential field used by RPP) the list of grasp assignments pruned as above vanishes at all the successors of the current configuration (call it $\boldsymbol{q}_{obj}$), the modified RPP resets the list attached to $\boldsymbol{q}_{obj}$ to contain all the possible grasp assignments at $\boldsymbol{q}_{obj}$ (as we proceed from the initial configuration). During a random motion (a motion intended to escape a local minimum of the potential), the list of grasp assignments is pruned but is constrained to never vanish. In the process of constructing τ_{obj}, the modified RPP may reset the grasp assignment list several times.

If successful, the outcome of RPP is a path τ_{obj} described as a series of configurations of $\mathcal{M}$, each annotated with a grasp assignment list. The path τ_{obj} is thus partitioned into a series of subpaths, each connecting two successive configurations. It defines as many transfer tasks as there are distinct grasp assignments associated with it. By construction, *for each such transfer task, there exists a transfer path satisfying the corresponding grasp assignment.* The number of regrasps along the generated path τ_{obj} is minimal, but RPP does not guarantee that this is the best path in that respect.

Details and comments: The condition that the same grasp assignment be possible at two neighboring configurations of $\mathcal{M}$ does not guarantee that the displacement of $\mathcal{M}$ can be done by a short (hence, collision-free) motion of the grasping arm(s). An additional test is needed when the set of grasps between two consecutive configurations is pruned. In our implementation, we assume that each arm has an inverse kinematics solution. Thus, an arm can attain a grasp with a finite set of different postures, determined by using the arm's inverse kinematics. We include the posture of each involved arm in the description of a grasp assignment. Hence the same combination of arms achieving the same grasp, but with two different postures of at least one arm, defines two distinct grasp assignments. Then a configuration of $\mathcal{M}$, along with a grasp assignment, uniquely defines the configurations of the grasping arms. The resolution of the grid placed across $\mathcal{C}_{obj}$ is set fine enough to guarantee that the motion between any two neighboring configurations of $\mathcal{M}$ results in a maximal arm displacement smaller than some prespecified threshold.

In addition, we make considerations for sliding grasps. Indeed, for some manipulation tasks, the object must be allowed to slide in the grasp of the arms to achieve the goal. For the grasp assignments in the grasp list, we identify their feasible neighbors and add them to the list. A neighboring grasp assignment is one where the grasp location and orientation is within some threshold distance from the original grasp assignment, and the same arm(s) and posture(s) is utilized. We choose a threshold distance that is small enough to ensure that it is feasible to slide between neighboring grasp assignments. By updating the grasp list in this manner, directions where a sliding grasp is necessary can then be explored during the search for an object path.

A transfer path could be obstructed by the arms not currently grasping $\mathcal{M}$. Dealing with these arms can be particularly complicated. In our current implementation, we assume that each arm has a relatively non-obstructive configuration given in the problem definition (in the system shown in Fig. 1, the given non-obstructive configuration of each arm is when they are held out to the side). Prior to a transfer path, all arms not involved in grasping $\mathcal{M}$ are moved to their non-obstructive configurations. The planner nevertheless checks that no collision occurs with them during the construction of τ_{obj}.

Perhaps the most blatant limitation of our approach is that it does not plan for regrasps at configurations of $\mathcal{M}$ where it makes contact with obstacles (as we said, τ_{obj} is computed free of collisions with obstacles). Since the object cannot levitate, we require that $\mathcal{M}$ be held at all times during regrasp. We assume that if $\mathcal{M}$ requires more than one arm to move, any subset of a grasp is sufficient to achieve static stability during the regrasp. For example, if a grasp requires two arms, any one of these arms, alone, achieves static stability, allowing the other arm to move along a transit path. An obvious example where this limitation may prevent our planner from finding a path is when the system contains a single arm; no regrasp is then possible.

RPP is only probabilistically complete [3]. If a path exists for $\mathcal{M}$, it will find it, but the computation time cannot be bounded in advance. Furthermore, if no path exists, RPP may run forever. Nevertheless, for a rigid object (as is the case for $\mathcal{M}$), RPP is usually very quick to return a path, when one exists. Hence, we can easily set a time limit beyond which it is safe to assume that no path exists.

RPP requires a postprocessing step to smooth the jerky portions of the path, due to the random walks. Other path planners could possibly be used in place of RPP.

Generation of transit paths: The transfer tasks identified as above can be organized into successive layers, as illustrated in Fig. 3. Each layer contains all the transfer tasks generated for the same subpath of τ_{obj}; the transfer tasks differ by the grasp assignment. Selecting one such task in every layer yields a series of transit tasks: the first consists of achieving the first grasp from the initial system configuration; it is followed by a possibly empty series of transit tasks to change grasps between two consecutive transfer tasks; the last transit task is to achieve the goal system configuration. Hence, it remains to identify a grasp assignment in each layer of the graph shown in Fig. 3, such that there exist transit paths accomplishing the corresponding transit tasks.

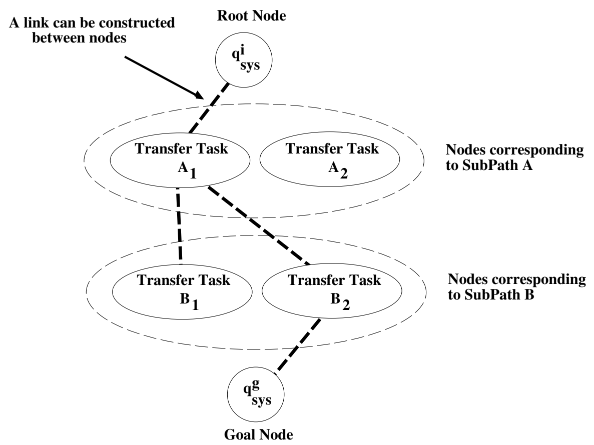

Figure 3. The directed and layered graph.

Assume without loss of generality that all arms are initially at their non-obstructive configurations. Our planner first chooses an arbitrary transfer task in the first layer. Consider the transit task of going from the initial system configuration to the configuration where the arms achieve the grasp assignment specified in the chosen transfer task, with $\mathcal{M}$ being at its initial configuration. The coordinated path of the arms is generated using RPP. If this fails, a new attempt is made with another transfer task in the first layer; otherwise, a transfer task is selected in the second layer. The connection of the system configuration at the end of the first transfer task to the system configuration at the start of this second transfer task forms a new transit task.

The transit task between two transfer tasks is more difficult to solve. Actually, the difficulty arises when an ungrasp/regrasp is required. If the initial and goal grasp assignments are neighbors, then we simply slide the grasp. To understand the difficulty of the case where an ungrasp/regrasp is required, imagine the situation where $\mathcal{M}$ is a long bar requiring two arms to move. Assume that the system contains only two arms and that the bar can be grasped at its two ends and at its center. Consider the situation where the bar is grasped at its two ends and the regrasp requires swapping the two arms. This regrasp is not possible without introducing an intermediate grasp. For example: arm 1 will ungrasp one end of the bar and regrasp it at its center (during this regrasp, arm 2 will be holding the bar without moving); then arm 2 will ungrasp the bar and regrasp it at the other end; finally, arm 1 will ungrasp the center of the bar and will regrasp it at its free end.

We address this difficulty by breaking the transit task between two transfer tasks into smaller transit subtasks. Each transit subtask consists of going from one grasp assignment to another in such a way that no two arms use the same grasp at the same time. In this process, we allow arms not involved in the first and last assignment to be used. We start with the first grasp assignment and generate all of the potential grasp assignments that may be achieved from it (assuming the corresponding transit paths exist). We generate the successors of these new assignments, and so on until we reach the desired assignment (the one used in the next transfer task). For each sequence that achieves this desired assignment, we test that it is actually feasible by using RPP to generate a transit path between every two successive grasp assignments. We stop as soon as we obtain a feasible sequence. The concatenation of the corresponding sequence of transit paths forms the transit path connecting the two considered transfer tasks. We then proceed to link to the next layer of transfer tasks.

When we reach a transfer path in the last layer, its connection to the goal system configuration is carried out in the same way as the connection of the initial system configuration to the first layer.

The result is an alternating sequence of transit and transfer paths that connects the initial configuration $\boldsymbol{q}^i_{sys}$ to the goal configuration $\boldsymbol{q}^g_{sys}$.

4 Human-Arm Kinematics

We now outline the method by which we determine the arm posture for a human arm given the position and orientation of its hand. We present the algorithm using the right arm for illustration purposes. We assume that the torso and the shoulder positions are given.

The algorithm is based on two results from neurophysiology. The first result has to do with decoupling the problem into two more manageable subproblems. Lacquaniti and Soechting have shown that the arm and wrist posture are for the most part independent of each other [14]. This allows us to decouple the problem into finding first, the forearm and upper arm posture to match the hand position, and then determining the joint angles for the wrist to match the hand orientation. This is exactly the approach taken in solving the inverse kinematics of a robot manipulator with six degrees of freedom and whose wrist joints intersect at a point [23].

It is also known in neurophysiology that the arm posture for pointing is mainly determined by a simple sensorimotor transformation model. Soechting and Flanders [29, 30] conducted experiments where the test subject was instructed to move the end of a pen-sized stylus to various targets in their vicinity. From this study, they have devised a model that determines the posture of the forearm and upper arm given the position of the end of the stylus. We use this model to determine the shoulder and elbow joint angles given the position of the hand. Then, determining the wrist joint angles is a simple additional step.

4.1 Arm posture

Using the sensorimotor transformation model of Soechting and Flanders [29, 30] we determine the arm posture given the location of the hand. To explain their model we first define some generalized coordinates.

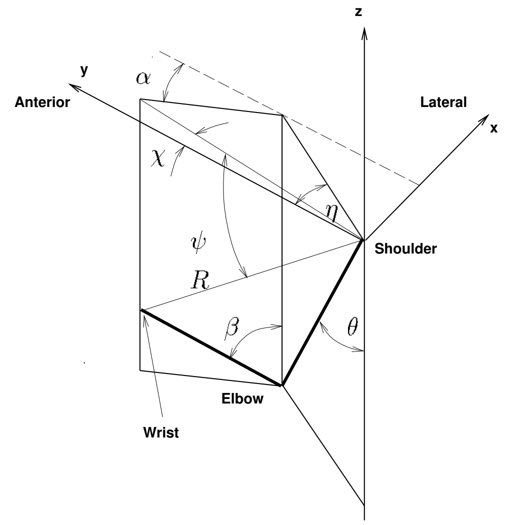

Figure 4. The parameters for the arm posture

Denote the coordinate frame centered on the shoulder as the *shoulder frame*. The x-axis is along the line that connects the two shoulders, the y-axis is parallel to the outward normal from the chest, and the z-axis points upwards towards the head. The parameters for the elevation and yaw of the upper arm are θ and η, respectively, and the parameters for the elevation and yaw of the forearm are β and α, respectively. The position of the wrist (or hand frame) is expressed in terms of the spherical coordinates, azimuth χ, elevation ψ, and radial distance R. The spherical coordinates are related to the cartesian coordinates of the shoulder frame by the equations

$$\left\{ \begin{array}{rc} R^2 = & x^2 + y^2 + z^2 \\ \tan\chi = & x/y \\ \tan\psi = & z/\sqrt{x^2+y^2} \end{array} \right. \qquad (1)$$

These arm parameters are illustrated in Fig. 4.

The sensorimotor transformation model suggests that the parameters for arm posture are approximated by a linear mapping from the spherical coordinates of the hand frame (actually, Soechting and Flanders report this mapping from the position of the end of the stylus held by the test subject, but we simplify this to the hand). The relation is

$$\left\{ \begin{array}{rc} \theta = & -4.0 + 1.10R + 0.90\psi \\ \beta = & 39.4 + 0.54R - 1.06\psi \\ \eta = & 13.2 + 0.86\chi + 0.11\psi \\ \alpha = & -10.0 + 1.08\chi - 0.35\psi \end{array} \right. \qquad (2)$$

where the units of measure are centimeters and degrees. Since this is only an approximation, plugging the arm posture parameters back into the forward kinematics of the arm results in a positional error of the hand frame. This is compensated for in the final stage of the inverse kinematics algorithm.

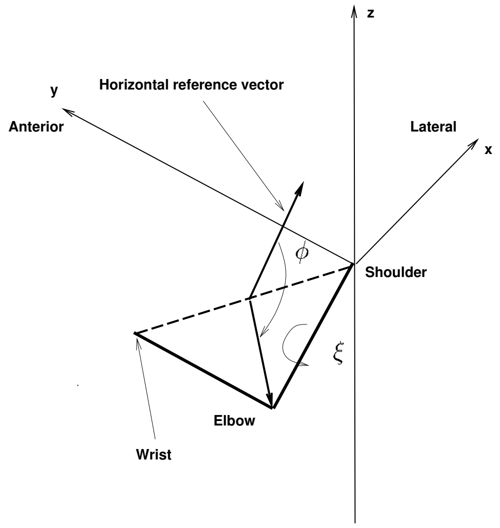

Figure 5. The elbow and shoulder rotation.

Once the generalized coordinates θ, β, η, and α are obtained, they are transformed into the four joint angles of the forearm and the upper arm. We check to see if they violate any of their limits. If they are within their limits we proceed to find the wrist joint angles. In the event that a limit is violated (an illegal posture), an adjustment phase is initiated.

For an illegal posture obtained from Eq. 2, it turns out that the joint angle ξ (the rotation around the upper arm as shown in Fig. 5) is the only one to violate its limits. We correct for this in the following manner. Consider a new set of generalized coordinates of the arm, consisting of the wrist position and the angle ϕ (the rotation of the elbow around the axis between the shoulder and wrist as shown in Fig. 5). By decreasing the value of ϕ from the initial illegal posture (moving the elbow upward), ξ will move back into the feasible joint range without ever changing the wrist position. Note that ξ is obtained by transforming the wrist position and ϕ into the joint angles of the arm.

To summarize, the arm posture is estimated in the following way:

1. The arm posture is obtained using the transformation of Eq. 2.
2. The values for ξ and ϕ are calculated.
3. If ξ violates its joint limit, then ϕ is decreased until the corresponding ξ satisfies its joint limit.

4.2 Wrist joints

The next step is to obtain the joint angles for the wrist.

We assume that the three wrist joint angles intersect at a point, consequently their values can be determined using basic inverse kinematics techniques. If the resulting wrist angles violate their limits, we adjust the posture of the arm until a feasible joint value is obtained. As before, this is achieved by incremently changing the angle ϕ (rotation of the elbow) until either a feasible solution is obtained or we have exhausted the possible values for ϕ. In the case where this step fails, we take the closest feasible answer to the desired hand orientation and proceed to the final adjustment step.

4.3 Final adjustment

The final step is to adjust the joint angles such that the exact position and orientation of the hand is obtained (recall that the sensorimotor transformation model leaves the hand position slightly off from the desired location). Let e be the 6×1 vector containing the position and orientation error of the hand frame, and q be the 7×1 vector of arm joint angles. The relation between these two vectors is

$$e = J(q)\dot{q} \tag{3}$$

where $J(q)$ is the 6×7 Jacobian matrix. By solving for $\dot{q}$ we can iteratively change the joint angles to zero the error.

This simultaneous equation has only six linear constraints, thus resulting in an infinite number of solutions. We remedy this situation by utilizing a solution of Eq. 3 expressed as

$$\dot{q} = \dot{q}_0 + r\,n \tag{4}$$

where $\dot{q}_0$, r, and n are respectively, an instance of $\dot{q}$, a scalar parameter, and the null space of Eq. 3. In this case, the null space is one dimensional and is expressed by the multiplication of the scalar r and its basis vector n. We employ a constrained optimization to select the $\dot{q}$ which minimizes its norm while satisfying the arm joint limits. The joint angles are then incrementally adjusted accordingly. This ensures that the modification to the hand position and orientation occurs with minimal joint movement. This procedure is iteratively applied until the desired position and orientation of the hand is obtained. The resulting joint angles comprise the posture of the arm for the given hand position and orientation.

The algorithm does not return an answer if the hand frame is out of the workspace of the arms, or if the method fails to correct for a set of joint angles which violates its limits.

5 Natural Motion

The multi-arm manipulation planner works regardless of whether the arms are human or robotic. In its most basic state, there is no consideration of producing realistic motions for the arms. Only kinematic constraints are recognized, for example: requiring the motion to be collision free and allowing the arms to grasp the movable object at all times.

However, for the purpose of computer animation the motions must appear natural. For robot arms, since they are undeniably artificial, there is no notion of what comprises natural movement. Though we may need to time parameterize their path to yield realistic animation (i.e. reasonable acceleration and deccelaration of the arms) there is no need to impose a naturalness constraint on the planner. In contrast, for human arms there is clearly a natural manner in which they move. Fortunately, neurophysiology gives us a reference to which we can compare our methods and results with experimental data on human motions. In this section, we discuss how our planner with the human-arm inverse kinematics satisfies applicable naturalness constraints one derives from neurophysiology.

Dynamics versus Kinematics: We make no consideration of dynamics in our planning. From the neurophysiological viewpoint this is not a problem. It is widely agreed upon that arm movement is represented kinematically [26]. It is then a postprocessing step, where the dynamics or muscle activation patterns are determined. With our method, by applying the appropriate time parameterization to the planned path [8, 1] one can emulate results found in neurophysiology. However, this is not done in our current planner yet.

Inverse Kinematics: The sensorimotor transformation model is derived from static arm postures. To verify that our inverse kinematics algorithm is applicable to manipulation motions, we consider the experimental results of Soechting and Terzuolo [27]. Their experiment is for curvilinear wrist motions in an arbitrary plane. They find that humans exhibit the following behavior:

1. The modulation in the elevation and yaw angles of the upper arm and forearm (θ, η, β, and α) are close to sinusoidal.

2. The phase difference between θ and β is $180°$, at least for elevation angles of the plane between $0°$ and $45°$.

3. The phase difference between α and θ is close to $180° - \sigma$, where σ is the slant parameter describing the curvilinear wrist motion.

We reproduced their experiment by tracing curvilinear wrist motions and generating the arm parameters using the human arm inverse kinematics algorithm. Our finding is that the computed and experimental values match quite nicely. We refer the reader to [13] for a detailed explanation. Furthermore, Soechting and Terzuolo find that for learned trajectories, their results for curvilinear wrist motions in a plane hold true in three-dimensional space [28]. We justify the use of our inverse kinematics algorithm for manipulation planning based on these experimental results. Note, we do not claim that the posture found by our algorithm is the *only one* that is natural. Clearly, there are other postures that humans assume depending upon the situation. For example, the redundant degree of freedom could be used to posture the arm such that it avoids obstacles. By having a library of different inverse kinematics algorithms for human arms, the planner could consider a variety of natural postures for a given hand position and orientation.

Point to Point Arm Motion: When the arms move to grasp or regrasp an object, the planning is done in the joint space of the arms. Since RPP is utilized to find the path, postprocessing is required to transform the collision free but jerky motion into a smooth one (the jerky motion is due to the random walks utilized by RPP). This smoothing is achieved by attempting to shorten the path by interpolating between its points. The result is essentially a piecewise joint interpolated path. Hollerbach and Atkeson speculate that the underlying planning strategy for such arm motions is a staggered joint interpolation [10]. Staggered joint interpolation is a generalization of joint interpolation introduced by Hollerbach and Atkeson to fit a greater range of experimental data. We have yet to implement a staggered joint interpolation scheme for the "smoother", but at present we are content with the arm motions produced thus far (i.e. joint interpolation appears to be a close enough approximation).

We were unable to find any studies on human regrasping. Thus, no comment can be made on how reasonable the planning strategy is from this viewpoint.

6 Experimental Results

We implemented the above approach in a planner written in C and running on a DEC Alpha workstation under UNIX. Experiments were conducted with a seated human figure and a robot arm. Each human arm has seven degrees of freedom, plus an additional nineteen degrees of freedom for each hand. The non-obstructive configurations for the human arms are ones in which they are held out to the side. The robot has six revolute joints and three degrees of freedom for the end-effector. The non-obstructive configuration for the robot is one in which it stands vertically. In addition, we seat the human on a swivel chair. By adding this extra degree of freedom, we allow the arms to access a greater region, and hence tackle more interesting manipulation tasks. The rotation of the chair tracks the object to keep it, essentially, in an optimal position with respect to the workspace of the arms [20].

Fig. 1 shows a path generated by the planner for the human arms to bring the glasses on the table to the head of the human figure. We specify that both arms should be used to manipulate the glasses (this is defined in the grasp set). Notice that during the regrasping phase, at least one arm is holding the glasses at all times. For this path it took one minute to identify the transfer tasks and an additional two minutes to complete the manipulation path. For the generation of τ_{obj}, the object's C-space was discretized into a $100 \times 100 \times 100$ grid. For the generation of the transit paths, the joint angles of the arms were discretized into intervals of 0.05 radians. The grasp set contained 212 grasps, yielding grasp assignment lists with up to 424 elements.

Fig. 6 shows a path generated by the planner for the human arms and the robot arm cooperating to manipulate a chess box. Having the different arms working together presents no difficulty to our planning approach. The planner simply needs to know the correct inverse kinematics algorithm to apply to each arm. For this example, in defining the grasp set, we specify two classes of grasps, one in which all three arms are used, and another in which only the two human arms are utilized. For this path it took about one and a half minutes to identify the transfer tasks and an additional two minutes to complete the manipulation path. The same discretizations as above were used. The grasp set of the box contained 289 grasps yielding grasp assignment lists with up to 2600 elements.

For manipulation planning with human arms, our current implementation is unable to plan motions where the arms are required to use their redundant degree of freedom to avoid obstacles. For example, a task where the arms must place an object deep into a tight box is almost impossible for our planner. The reason is simply that the sensorimotor transformation model does not consider obstacle avoidance. To tackle this class of problem, we will need to devise another inverse kinematics algorithm that *does* utilize the redundant degree of freedom for the purpose of avoiding obstacles. Again, some sort of naturalness constraint would need to be satisfied.

For these examples, in computing the transit paths RPP uses the sum of the angular joint distances to the goal configuration as the guiding potential. Note that the motion of the fingers for the human and the robot are considered in the transit paths (in moving from one grasp to another the fingers may change their posture). In computing τ_{obj} RPP uses an NF2-based potential with three control points [16]. In finding both the transit paths and τ_{obj}, we limit the amount of computation spent in RPP to three backtrack operations [16], after which the planner returns failure. Failure to find τ_{obj} results in the immediate failure to find a manipulation path. Similarly, a failure to find transit paths to link together the layers of transfer paths results in a failure to find a manipulation path. The time for the planner to report failure depends on the problem, with some examples ranging from 30 seconds to a few minutes. The collision checking algorithm utilized is that of Quinlan [24].

7 Conclusion

We have presented a novel approach for solving the complicated multi-arm manipulation planning problem. Our approach embeds several simplifications yielding an implemented planner that is not fully general. However, experiments with this planner show that it is quite reliable and efficient in finding manipulation paths, when such paths exist, making it suitable as part of an interactive tool to facilitate the animation of scenes. We believe the robust nature of the planner is the result of careful consideration of the general manipulation problem, the introduction of reasonable simplifications, and the appropriate utilization of the efficient randomized path planner.

We have also presented a new inverse kinematics algorithm for human arms based on neurophysiological studies. This algorithm, in conjunction with the planner, automatically generates natural arm motions for a human figure manipulating an object.

In the future we hope to include the ability to regrasp the movable object by having the arms place it in a stable configuration against some obstacles. We also plan to use existing results to automatically compute the grasp set of an object from its geometric model. The technique described in [11] to deal with multiple movable objects in a 2D space should also be applicable to our planner. Furthermore, we hope to time parameterize the motion paths to yield realistic velocities, by implementing one of the many such algorithms for robot and human arms [8, 1, 4]. Finally, we hope to explore and devise other inverse kinematics algorithms for the arms, as well as incorporating twisting and bending of the torso. Ultimately, we aim to create a task-level animation package for human motions.

References

[1] C.G. Atkeson and J.M. Hollerbach, "Kinematic Features of Unrestrained Arm Movements," *MIT AI Memo 790*, 1984.

[2] D. Baraff, "Analytical Method for Dynamic Simulation of Non-Penetrating Rigid Bodies," *Computer Graphics*, 23(3), 1989, pp. 223-232.

[3] J. Barraquand and J.C. Latombe, "Robot Motion Planning: A Distributed Representation Approach," *Int. J. Robotics Research*, 10(6), December 1991.

[4] J.E. Bobrow, S. Dubowsky, J.S. Gibson, "Time-Optimal Control of Robotic Manipulators Along Specified Paths," *International Journal of Robotics Research*, Vol. 4, No. 3, 1985.

[5] A. Bruderlin and T.W. Calvert, "Goal-directed, dynamic animation of human walking," *Computer Graphics,* Vol. 23, No. 3, 1989, pp.233-242.

[6] W. Ching and N. Badler, "Fast motion planning for anthropometric figures with many degrees of freedom," *Proc. 1992 IEEE Int. Conf. on Robotics and Automation,* 1992, pp.2340-2345.

[7] P. Ferbach and J. Barraquand, *A Penalty Function Method for Constrained Motion Planning*, Rep. No. 34, Paris Research Lab., DEC, Sept. 1993.

[8] T. Flash and N. Hogan, "The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model," *MIT AI Memo 786*, 1984.

[9] J. Hahn, "Realistic Animation of Rigid Bodies," *Computer Graphics*, 22(4): 1988, pp. 299-308.

[10] J.M. Hollerbach and C.G. Atkeson, "Deducing Planning Variables from Experimental Arm Trajectories: Pitfalls and Possibilities," *Biological Cybernetics,* 56(5), 1987, pp. 279-292.

[11] Y. Koga, T. Lastennet, J.C. Latombe, and T.Y. Li "Multi-Arm Manipulation Planning," *Proc. 9th Int. Symp. Automation and Robotics in Construction*, Tokyo, June 1992.

[12] Y. Koga, PhD. dissertation (in preparation)

[13] K. Kondo, "Inverse Kinematics of a Human Arm," (in preparation)

[14] F. Lacquaniti and J.F. Soechting, "Coordination of Arm and Wrist Motion During A Reaching Task," *The Journal of Neuroscience,* Vol. 2, No. 2, 1982, pp399-408.

[15] J.P. Laumond and R. Alami, *A Geometrical Approach to Planning Manipulation Tasks: The Case of a Circular Robot and a Movable Circular Object Amidst Polygonal Obstacles*, Rep. No. 88314, LAAS, Toulouse, 1989.

[16] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[17] P. Lee, S. Wei, J. Zhao, and N.I. Badler, "Strength guided motion," *Computer Graphics,* Vol. 24, No. 4, 1990, pp.253-262.

[18] J. Lengyel, M. Reichert, B.R. Donald and D.P. Greenberg, "Real-Time Robot Motion Planning Using Rasterizing Computer Graphics Hardware," *Computer Graphics,* Vol. 24, No. 4, 1990, pp.327-335.

[19] T. Lozano-Pérez, "Spatial Planning: A Configuration Space Approach," *IEEE Tr. Computers,* 32(2), 1983, pp. 108-120.

[20] E.J. McCormick and M.S. Sanders, *Human Factors in Engineering and Design*, McGraw-Hill Book Company, New York, 1982.

[21] M. McKenna and D. Zeltzer, "Dynamic simulation of autonomous legged locomotion," *Computer Graphics,* Vol. 24, No. 4, 1990, pp.29-38.

[22] J. Pertin-Troccaz, "Grasping: A State of the Art," in *The Robotics Review 1,* O. Khatib, J.J. Craig, and T. Lozano-Pérez, eds., MIT Press, Cambridge, MA, 1989, pp. 71-98.

[23] D. Pieper and B. Roth, "The Kinematics of Manipulators Under Computer Control," *Proceedings of the Second International Congress on Theory of Machines and Mechanisms*, Vol. 2, Zakopane, Poland, 1969, pp. 159-169.

[24] S. Quinlan, PhD. dissertation (in preparation)

[25] M.H. Raibert and J.K. Hodgins, "Animation of Dynamic Legged Locomotion," *Computer Graphics,* Vol. 25, No. 4, 1991, pp.349-358.

[26] A.M. Smith et al., "Group Report: What Do Studies of Specific Motor Acts Such as Reaching and Grasping Tell Us about the General Principles of Goal-Directed Motor Behaviour?'," *Motor Contro: Concepts and Issues*, D.R Humphrey and H,J, Freund, eds., John Wiley and Sons, New York, 1991, pp. 357-381.

[27] J.F. Soechting and C.A. Terzuolo, "An Algorithm for the Generation of Curvilinear Wrist Motion in an Arbitrary Plane in Three Dimensional Space," *Neuroscience,* Vol. 19, No. 4, 1986, pp. 1393-1405.

[28] J.F. Soechting and C.A. Terzuolo, "Organization of Arm Movements in Three Dimensional Space. Wrist Motion is Piecewise Planar," *Neuroscience,* Vol. 23, No. 1, 1987, pp. 53-61.

[29] J.F. Soechting and M. Flanders, "Sensorimotor Representations for Pointing to Targets in Three Dimensional Space," *Journal of Neurophysiology,* Vol. 62, No. 2, 1989, pp.582-594.

[30] J.F. Soechting and M. Flanders, "Errors in Pointing are Due to Approximations in Sensorimotor Transformations," *Journal of Neurophysiology,* Vol. 62, No. 2, 1989, pp.595-608.

[31] J.F. Soechting, "Elements of Coordinated Arm Movements in Three-Dimensional Space," *Perspectives on the Coordination of Movement,* edited by S.A. Wallace, Elsevier Science Publishers, Amsterdam, 1989, pp.47-83.

[32] P. Tournassoud, T. Lozano-Pérez, and E. Mazer, "Regrasping," *Proc. IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, 1987, pp. 1924-1928.

[33] B. Webber, N. Badler, F.B. Baldwin, W. Beckett, B. DiEugenio, C. Geib, M. Jung, L. Levinson, M. Moore, and M. White, "Doing what you're told: following task instructions in changing, but hospitable environments," *SIGGRAPH '93 Course note 80 "Recent Techniques in Human Modeling, Animation and Rendering",* 1993, pp. 4.3-4.31.

[34] G. Wilfong, "Motion Planning in the Presence of Movable Obstacles," *Proc. 4th ACM Symp. Computational Geometry*, 1988, pp. 279-288.