Augmenting Human Performance in Motion Planning Tasks – the
Configuration Space Approach *

I. Ivanisevic and V. Lumelsky
Robotics Lab, University of Wisconsin-Madison
Madison, Wisconsin 53706, USA
igor@cs.wisc.edu

Abstract

This paper is concerned with the development of methods that can be used to augment human performance in three degree of freedom (DOF) motion planning tasks. Previous experimental studies have shown that human performance in such motion planning tasks is inadequate, especially when complex objects and/or environments are involved. Our approach is based on the use of Configuration Space (C-Space) to simplify the motion planning task from that of moving a complex object in work space (W-space) to moving a point in a C-space maze. Tools are developed which allow the user to easily grasp all the information contained in 3D C-space and use it to solve the task at hand.

1 Introduction and Prior Work

Consider a complex assembly with thousands of parts – say, an aircraft engine. The engine has been designed using modern CAD tools; the corresponding database resides on a computer disk. At this stage, before a hardware prototype has been built, the designer would like to assess the feasibility of removing a certain block or part from the assembly. Answering this question, and perhaps optimizing the removal process is essential for the future maintenance program. For example, if replacing the part takes more than 30 minutes, doing it at the airport gate is not feasible, the aircraft must be put in the hangar, and the whole operation becomes significantly costlier. A satisfactory motion sequence can be transferred to the maintenance crew instruction manual. There are many similar

*This work was supported by the Office of Naval Research Grant N000149710871.

object manipulation problems where one attempts to move an object from one position to another in a crowded space, such as in computer-assisted surgery, teleoperation of shuttle arm manipulators or safety and landmine handling vehicles, animation of virtual reality scenes etc.

The formal question here is one of path existence – is there a path with translations and rotations that will take the object from its current to a desired position? In this general formulation, it is a motion planning problem with 6 degrees of freedom (DOF), and it is known to be NP-hard. With today’s algorithms, solutions to such problems may require hundreds of hours of top speed computer stations, or may not be practical at all [1].

Prior attempts to solve this problem have been based primarily on techniques of motion planning with complete information (also called the Piano Mover’s Problem [2]). Since its direct application carries prohibitive computational costs, researchers focused on ways of limiting the number of options of object’s paths analyzed. One popular method that emerged is the randomized search planning [3]. Compared to the exhaustive search of the Piano Mover’s method, random search algorithms achieve computational savings by testing only a (semi-randomly chosen) fraction of possible path options. The direction of motion is guided by a potential field designed to produce a global extremum at the target position. As a variation of the procedure, the search may be shifted from the work space to the task’s configuration space (C-space) [4]. Since in general a potential field has local extrema, methods are designed to overcome those extrema on the way to the global extremum.

While offering significant computational savings compared to the direct Piano Mover’s approach, randomized search techniques suffer from this cir-
cular problem: (a) if the algorithm does not find a path, this does not mean a path does not exist; (b) to improve chances of not losing a legitimate path, one has to increase the random search space, which increases the required time. As such approaches rely on a fully automated processing, they leave out an important source of efficiency – on-line help by a human operator. While our previous studies [5, 6] confirmed the industry experience that humans have difficulties with spatial reasoning tasks involving motion planning for complex objects, they are quite good in assessing the global situation. Many path options diligently considered by an automatic system – say, an attempt to pass through a dead-end corridor or a passage that is too narrow – would look silly to a human observer. What is missing in an automatic system is a poorly-understood powerful human ability “to see the big picture”, “to see what makes sense”.

2 Motivation

It is this complementarity between the human and machine intelligence that this work intends to exploit. What is envisioned is a human-centered system in which both types of intelligence work on the task in a team-like manner, simultaneously, and comfortably within the human time scale (say, from seconds to 10-20min rather than 100 hours); we will call this a real time operation.

Previous studies [5, 6] demonstrated the fact that we humans can operate quite efficiently in a very complex environment as long as the body to be moved is, or can be taken “in the big picture” to be, a point, so there is no need to analyze simultaneous collisions at the body’s different parts. Also, human performance in motion planning is better when dealing with translations as compared to rotations. When forced to deal directly with kinematics, rotations, and multiple contacts, human subjects tend to try many path options that would immediately look useless to a good planning algorithm.

Of such observations comes an idea that an appropriate interface that transforms a problem with translations and rotations into one with translations only, while at the same time transforming multi-point contacts to single-point contacts, would have a good chance of improving human performance. To test the feasibility of this idea, we will apply it to a 3 DOF version of the above problem - motion planning for an object in the plane with translation and rotation. If successful, such an approach can already satisfy some interesting real-world problems, but it would still fall short of tasks like an aircraft engine part manipulation problem which is a 3D problem with 6 DOF (three translations and three rotations). This next level of complexity would require a more active partnership between the operator and a motion planning algorithm – some possible directions of future research in this area are discussed in Section 6.

The visual human-computer interface described in this paper transforms the task of motion planning for a complicated object in the plane (the work space (W-space) control) to one of control of a point in the corresponding configuration space (C-space). Since, for these types of tasks, the information about the environment already exists in a database, computing the corresponding task configuration space is not a problem. Our previous work [5, 6] successfully applied a similar transformation for the cases of teleoperation of 2 DOF and 3 DOF robot arm manipulators which were problems that only involved rotations. In this paper, we demonstrate that a similar transformation can successfully be applied to problems which involve both translations and rotations.

The task considered in this paper is discussed in Section 3, the proposed C-space based interface appears in Section 4, and results are given in Section 5. A summary of the paper’s contribution is found in Section 6.

3 The Task

Consider an example shown in Figure 1. The task is to move a machine (an object shown in gray) out of the room on the factory floor – from “Start” to somewhere around the “Target”. Note the complex geometry of objects in the room – the database of the problem is rather extensive. This 2D task presents a 3D motion planning problem with two translations and one rotation.

Task Configuration Space (C-space). The object’s position in this task is uniquely defined by the triple \( (x, y, \theta) \), where \( x \) and \( y \) refer to the Cartesian coordinates of the position of the object’s center on the factory floor, and \( \theta \) describes
the object’s counter-clockwise rotation in the plane about its center.

The C-space for this task is a 3D space of the three variables \((x, y, \theta)\). One can use different algorithms to compute this C-space - here, C-space has been computed in a “brute-force” fashion: the object is placed in all possible (digitized) configurations in W-space; the set of all configurations in which the object touches one or more obstacles form the surface(s) of C-space obstacles. The complexity of this procedure is \(O(n^3)\) for this 3 DOF task. With the resolution of about 1% of the variable’s full range along each degree of freedom, the entire computation takes less than 5 min on a modern workstation. Once computed, the C-space for this task can be saved and need not be computed again unless changes to obstacle locations are made.

4 W-space vs. C-space planning

If we rely on the human to solve this task, two separate methods of controlling the object can be envisioned. In the traditional approach, the human user might be asked to find the path for the object in W-space. The user is provided with e.g. a mouse/keyboard interface and a view of the factory floor containing the object, similar to the one shown in Figure 1. For example, the mouse may be used to control the translation of the object while the keyboard controls its rotation. The task is considered complete when the object has been moved from its starting configuration to the neighborhood of the target configuration. This approach suffers from the aforementioned deficiencies of human spatial reasoning when planning the motion of complex objects. The user may have difficulty finding a path that solves this task but, if this is the case, they cannot be sure such a path does not exist. Further, if a path is found, the user will probably have a very hard time deciding if more efficient paths may exist.

Alternatively, C-space feedback and control can be considered. As explained in the previous section, transforming the problem of moving a complex object in W-space to that of moving a simple point in C-space greatly simplifies the task from the standpoint of the human user. The question becomes one of presenting this 3D C-space maze to the human in a way that would allow them to quickly decide if a path is feasible (and produce such a path, if necessary - e.g. for a manual).

If we simply present the user with a screen projection of 3D C-space, they will not benefit from all the information contained in this space. The ability to rotate the view of the C-space and show projections of important features (e.g. current position and orientation of the object) may help somewhat, but the fundamental problem remains: the viewer’s position is part of 3D C-space and, therefore, the user’s view is obstructed by C-space obstacles in the environment. This problem is compounded by the fact that 3D C-space obstacles tend to be cave-like structures with multiple entrances, branches and dead-ends inside [7].

Attempting a strategy which would require the user to lead the point “inside” such structures, with no knowledge of which entrance leads to the target and which to a dead-end, is likely to negate the human ability to see “the big picture” and degenerate into an exhaustive search of the insides of C-space obstacles. To alleviate this problem, we propose a second mapping which transforms the 3D C-space into a discontinuous 2D space of “sliced” C-space obstacles. This slicing was developed in [6] for use in teleoperation of robot arm manipulators. It works as follows: assume the two axes of the 3D C-space corresponding to variables \(x\) and \(y\), have a length of 100 units.
Figure 2: Slicing of the 3D C-space of the task in Figure 1. S and T represent start and target configurations from that Figure. C-space obstacles are shaded regions in each slice.

Further, assume the axis corresponding to variable $\theta$ has a length of $2\pi$. The former two axes have a “range” obstacle automatically placed at 0 and 100 - this ensures that the object cannot wander outside of the factory floor (these range obstacles span the $(y, \theta)$ and $(x, \theta)$ planes, for the given values of $x$ and $y$, respectively). The $\theta$ axis has no such range obstacles - the object can rotate more than $2\pi$ radians. Any values of $\theta$ outside the $[0, 2\pi]$ range are reduced to $[0, 2\pi]$ through simple subtraction. Assume the axis corresponding to $\theta$ is positioned on the screen in the “vertical” direction (i.e. up and down, as opposed to left-right or away-towards the viewer). The C-space cube is now sliced by horizontal planes, top down, along the $\theta$ axis, in increments of $\Delta \theta$.

The result is a number of $(x, y)$ squares (of side length 100), each representing a slice of C-space for a fixed value of $\theta$. Each square contains slices of regular C-space obstacles as well as the range obstacles (placed at 0 and 100 along both the x and y axes). These squares form the basis of the interface the operator is to use in C-space control. The number of squares is controlled by the size of the slice increment, $\Delta \theta$, which is determined by the user as a tradeoff between screen size, convenience, and desired resolution (more slices result in better resolution). Of course, if the C-space was computed using a resolution of $1\%$ along each axis, choosing a $\Delta \theta$ of less than $\pi / 50$ radians would not make sense - it would exceed the resolution of the C-space being cut.
Additional rules that connect each square with its neighbors must be introduced. Squares are placed from left to right in each row; going from a square to its neighbor to the right corresponds to adding \( \Delta \theta \) to \( \theta \) (i.e., rotating the object counterclockwise by \( \Delta \theta \) radians). The rightmost square in row \( n \) (assuming rows are numbered from the top of the screen down) is a neighbor of the leftmost square in row \( n + 1 \); i.e., the \( \theta \) value of the former is \( \Delta \theta \) bigger than that of the latter. The upper left corner in the figure corresponds to \( \theta = 0 \), the C-space floor; the bottom right corner corresponds to \( \theta = 2\pi - \Delta \theta \). These lower and upper squares are neighbors since the object is allowed to rotate more than \( 2\pi \) radians.

One might ask why \( \theta \) is chosen as the direction of slicing. The decision is somewhat arbitrary but does present some advantages. In particular, consider a case where the \((x,y)\) position of the center of the object places it well inside some obstacle. Regardless of which value of \( \theta \) is chosen (i.e., how the object is rotated), the object will be in contact with the obstacle and a C-space virtual obstacle will exist for this configuration. This case is observed as a C-space obstacle that spans the entire range of \( \theta \). Since the \( \theta \) axis is placed in the vertical direction, a set of such obstacles in close proximity to one another results in a “pillar” shape. The main benefit of exploiting this “pillar” property by slicing along the \( \theta \) axis lies in the fact that neighboring horizontal cross-sections of the vertical columns are very similar in appearance (see Figure 2). This similarity provides the human user with a basis for comparison when considering the transition from square to square that would not be present if neighboring squares contained vastly different mazes.

Figure 2 shows a sliced C-space of the task in Figure 1. Dark areas represent C-space obstacles (both range and otherwise) while white space stands for collision-free configurations. S and T mark the start and target configurations corresponding to those in Figure 1. The interface for C-space control allows the user to move the arm C-point within the current square, or between neighboring squares, or to try to jump over many squares - this will work if the straight line in C-space between the two jump points does not cross any obstacles. Zoom feature is also provided - e.g. to allow the user to study a square more carefully. Typically the user might look for half-minute or so at the screen, mentally formulate a plan, and then set to execute it. One may notice that the sliced C-space for this task is very crowded - this is somewhat expected given the large number of obstacles in the task’s W-space. Fortunately, crowded C-space actually makes the task much easier to solve using the sliced C-space interface - one need only examine the (few) areas of free space while looking for a solution.

5 Results

Figure 3 shows a solution to the task in Figure 1 translated back to W-space but obtained using sliced C-space control, Figure 2. Examining Figure 2, the user determined that the solution to this task involves two steps. Since both S and T are located in “holes” fully enclosed by C-space obstacles on their respective squares, the first task is to locate a square where the two “holes” are connected. Once this is accomplished, the point can be guided from the “hole” containing S to the one which surrounds T, at which point it can easily be led back through a sequence of squares to the square containing point T.

Let’s number the squares from 1 to 100, starting at the top left square and going left-right and top-
bottom, similar to reading a page in a book. One can notice that on square 20 (rightmost square in the second row) the two “holes” become connected via an opening. Hence, the operator guides the point to this opening and into the second “hole”. Notice that, on the way to square 20, the point “drops down” a row when going from square 10 to square 11 and wraps around from the rightmost square in row 1 (square 10) to leftmost square in row 2 (square 11). The rest of the solution involves backtracking from square 20 to square 1. As mentioned in the previous section, square 1 is connected to square 100 (due to the “wraparound” property along the \( \theta \) axis), so we can guide the point directly from square 1 to square 100 and on to T.

This entire procedure takes no more than a minute or two for an operator familiar with the special rules of sliced C-space. If the operator simply wanted to determine whether a solution exists, the procedure would take no more than a few seconds. Compare this result to the operator’s performance in W-space, where they might take 10 minutes to find a path less efficient than the one obtained using the C-space interface. Previous studies that compared sliced C-space and W-space operation suggest a significant performance improvement can be expected (on the order of 2-4 times better performance) [7].

6 Conclusion

This paper discusses an approach that improves a person’s ability to solve motion planning tasks involving an object in the plane by transforming the problem from traditional work space to configuration space. The improvements come as a result of simplifying the problem of collision analysis, which can be very challenging for human spatial reasoning in W-space. In the proposed 2D sliced version of the 3D task C-space, the collision analysis problem becomes trivial and the human user can concentrate on efficient “global planning”. Despite the strange appearance of the sliced C-space and a few unintuitive rules of motion within it, human operators quickly learn to perform admirably using the proposed interface.

The results presented in this paper show that sliced C-space control can successfully be applied to motion planning problems which involve both translations and rotations (our previous work considered only rotations). Future work using the C-space control method shows potential to solve the general 6 DOF motion planning problem. What is needed is a 3 DOF algorithm that can work in conjunction with the operator using the sliced C-space interface. For the problem presented in this paper, this division of responsibility might involve the human operator controlling the 3 translational DOF (since humans are better at handling translations) while the algorithm automatically takes care of the rotations.

References


