Optimizing Robot Placement for Visit-Point Tasks *

Yong K. Hwang and Peter A. Watterberg
Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

We present a manipulator placement algorithm for minimizing the length of the manipulator motion performing a visit-point task such as spot welding. Given a set of points for the tool of a manipulator to visit, our algorithm finds the shortest robot motion required to visit the points from each possible base configuration. The base configuration resulting in the shortest motion is selected as the optimal robot placement. The shortest robot motion required for visiting multiple points from a given base configuration is computed using a variant of the traveling salesman algorithm in the robot joint space and a point-to-point path planner that plans collision-free robot paths between two configurations. Our robot placement algorithm is expected to reduce the robot cycle time during visit-point tasks, as well as speeding up the robot set-up process when building a manufacturing line.

1 Introduction

A significant number of tasks in manufacturing requires the robotic manipulator to visit a number of points with its tool tip. Some of these tasks include spot welding, sampling, marking and inspection. Traditionally, the position and orientation of the robot base (base configuration, BC) are determined using the reachability envelop and a reachability test. In other words, the user places the robot so that the points to visit with the tool (tool configurations, TC) fall within the reachability envelop of the robot. An inverse kinematics routine is then used to ensure that the robot can in fact reach each TC with its tool tip without colliding with objects in the workcell. This process can be repeated for a number of BCs to minimize the total cycle time of the robot performing the visit-point task. Development of robot simulation software [12, 13] has made it easier to perform the procedure described above. Rather than moving the physical robot, a graphical robot is manipulated in a simulated environment to check reachability. Still, this is a time consuming process and does not guarantee the optimality of the robot base configuration in a systematic manner.

This paper presents an algorithm that uses a brute-force search in the BC space, a collision-free motion planner and a traveling salesman algorithm in the joint space to guarantee the optimality of the BC. During the process, it also computes the joint angles necessary to visit each TC and the optimal ordering of the TCs. The Sandros motion planner [2] that finds a collision-free path between two robot configurations plays a central role in our BC determination algorithm. Because of the off-line nature of the problem and the significance of reducing the robot cycle time in manufacturing, hours or even days of computation time can be justifiably spent to carry out the brute-force search. Our algorithm offers two benefits to manufacturing processes. First, it minimizes the robot path length which is directly related to the cycle time and the throughput of a manufacturing line. Second, it reduces the set-up time of manufacturing lines by computing optimal robot placements automatically. This in turn shortens the product development cycle, giving manufacturers a competitive edge.

2 Previous Work

Research on BC optimization has not received as much attention as other planning problems in robotics for two reasons. First, the traditional method of moving the robot and checking the reachability has worked to an acceptable degree. Experienced robot technicians have been able to place the robot near the optimal base configuration with a reasonable number of trials. Second, the lack of an automatic motion planner that finds a short, collision-free motion has limited the simulation software to checking the reachability only. Finding the optimal robot BC requires full consideration of paths between points, as well as planning the optimal ordering of the points to visit. Practical algorithms for path planning and sequencing have not been available until recently [2, 5].

There has been some work done in automating workcell design and robot placement strategies with different emphases or methods. A semi-automatic method of generating an assembly workcell layout is presented in [10]. A workcell designer first generates a rough layout of robots and other production resources. The visit points for the robot tool are generated from an assembly plan and part locations and reachability of these points is checked. A numerical optimization routine then assesses the quality of the layout and computes a new layout. This process is iterated until a satisfactory layout is found. Paths between points are found after the layout is determined. Their work is aimed at developing a layout algorithm for an overall assembly workcell. In contrast, our algorithm specifically determines optimal robot base configurations, with the optimality being the *actual, not estimated*, and *collision-free* path length of

*This work has been performed at Sandia National Laboratories and supported by the U.S. Department of Energy under Contract DE-AC04-94DP85000.

DISCLAIMER

the robot. Determining the optimal robot placement for a robot performing a transfer movement between two points is considered in [4]. This algorithm uses a gradient descent approach rather than a brute-force search to find the optimal robot placement. This algorithm minimizes the robot execution time by taking into account the joint acceleration, whereas ours minimizes the length of the *collision-free* path that visits *multiple* points. Work on selecting robots and deploying them to proper work cells for a CIM system is presented in [3], but it is concerned with layout design of a whole manufacturing line from an industrial-engineering perspective.

The main contribution of this paper is an algorithm to find the exact robot placement that minimizes the total joint path length visiting a set of tool configurations. This paper is organized as follows. Our algorithm is described in Section 3. In Section 4 the performance results are illustrated in a spot-welding application, and our conclusions and future research areas are summarized in the final section. We use the terms *robot* and *manipulator* interchangeably. We define *robot configuration* to be the set of robot joint angles and *tool configuration* to be the position and orientation of the robot tool tip. We define *dof* to be the number of degrees of freedom of the robot.

3 Algorithm

Our robot placement (RP) algorithm is based on the visit-point (VP) algorithm presented in [5]. Given a set of *task* points, the VP algorithm finds a short, collision-free motion that visits the task points with the tool tip. The RP algorithm places the robot base in all possible configurations given by the user and runs the VP algorithm to obtain the shortest motion visiting the points in any order. The robot base configuration that results in a path with the smallest length in the joint space is selected as the optimum. See Figure 1 for an illustration of our algorithm. We now describe the three major components of the RP algorithm in detail.

3.1 Inverse kinematic solutions

Given a task point, there are usually multiple robot configurations (called inverse kinematic solutions, IKS) to reach that point. There are two reasons for multiple configurations. First, many visit-point tasks can be performed from multiple tool configurations. For example, spot welding tasks and drilling operations allow variations in the roll angle of the welding tip and drill bit, respectively. We handle this case by computing the IKS for each roll angle at a dense enough interval. Second, even if the tool configuration is completely specified at the task point, there is a finite number of IKS for a 6-dof robot, and usually an infinite number of solutions if the robot has redundancy, i.e., dof>6. If the number of solutions is finite, we compute all solutions. If infinite, we sample solutions with an approximate uniform distribution over the solution manifold. One such method is presented in [7]. Once the IKS are found, collision checking is performed to prune out those that involve collisions between the robot and objects in the workcell.

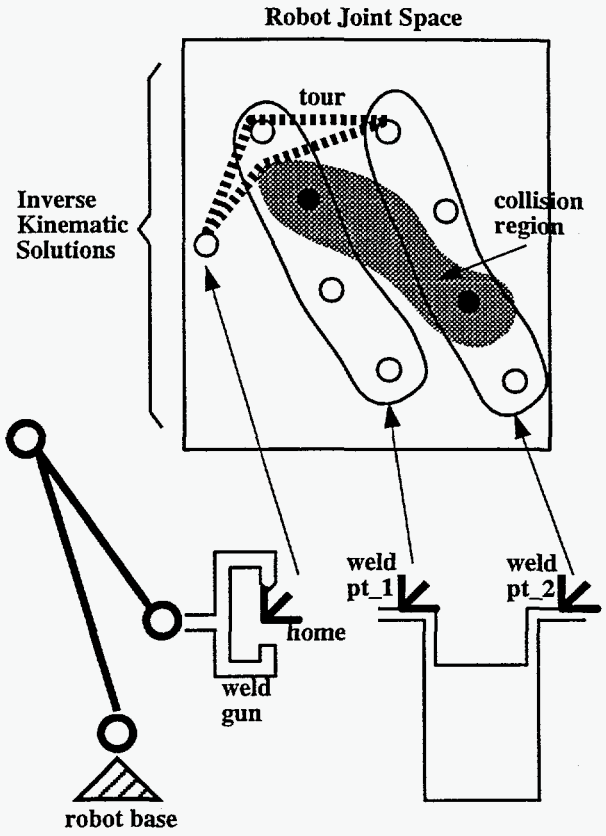

Figure 1: Starting from its home position, a robot is to spot weld two points. Each weld point has four inverse kinematic solutions, one of which causes collisions between the robot and the object. The shortest tour for the welding task is shown in broken lines in the joint space. The tour length can be further minimized by varying the robot base configuration.

3.2 VP algorithm

The VP algorithm is presented in [5] and is summarized here for the self-containment of this paper. Given a set of task points that the robot is to visit with its tool tip and the associated IKS for each task point, a variation of the traveling salesman problem (TSP), which we have named *Group* TSP (GTSP), is formulated. The major difference is that in GTSP, an admissible tour is required to include only one of the IKS for each task point. In addition, the VP problem has unknown edge costs, since the actual length of a collision-free path between two robot configurations is available only after a fairly expensive application of a point-to-point path planner. If the number of task points and associated IKS are small, it is possible to find the length of a collision-free path between every pair of IKS. However, in our implementation we use the lower bound of the path length, namely, the straight-line distance between two configurations in joint space to find a short tour. We then compute the actual length of the collision-free path corresponding to the tour by applying a point-to-point motion planner to each segment of the tour.

Our VP algorithm uses a variation of the Nearest Insertion Algorithm [9] to solve the GTSP component and uses the Sandros planner [2] to solve the point-to-point motion planning problem. Our Nearest Insertion Algorithm works as follows. Starting from the home position of the robot, we find a shortest tour that includes one IKS for each task point. Starting with the partial tour that includes only the home position, the IKS closest to the partial tour among the IKS corresponding to the unvisited task points are iteratively added to the partial tour until one IKS from each task point is included. When adding an IKS to the partial tour, it is inserted so as to minimize the length of the resulting partial tour (nearest insertion). The Sandros planner is then used to compute the collision-free path along this tour, and thus the actual tour length. We repeat this process until each of the IKS is included in at least one of the tours. The tour with the shortest path length is chosen as the solution tour. Of course, if the computation resource is available, we would resort to an exhaustive search for the GTSP component. The complexity of the GTSP is $O(I^T Complex(TSP))$, where T is the number of the task points, I is the number of IKS for each task point, and $Complex(TSP)$ is the complexity of the regular traveling salesman problem. The regular TSP is NP-complete and $O(n^2 2^n)$ time is required to obtain a near-optimal solution [9]. Thus, for any practical number of T and I, the complexity of GTSP prohibits exhaustive search. Our algorithm gives a systematic framework to compute approximately optimal solutions by decomposing the problem into point-to-point path planning, inverse kinematics and GTSP. Any one of the components can be replaced with a more efficient or sophisticated algorithm, depending upon the available computing resource.

3.3 Robot base configurations

The user first specifies the degrees of freedom of the robot-base configuration and the associated range for each dof. The dof is usually 3 (x, y, θ) or 4 (x, y, z, θ). Assuming we discretize each dof with 10 points, there are 10^3 or 10^4 possible robot base configurations. The 10-point discretization is not coarse since the user usually has a rough idea of the optimal robot-base location. Assuming that the shortest path for one base configuration is computable in 1 minute, the RP algorithm will take on the order of several days. Since our algorithm will be used for off-line applications, e.g., determining the base location of a spot-welding robot once at the installation time of a manufacturing line, days of computation time is acceptable. However, this time may be reduced to several hours by using a network of workstations or a supercomputer.

After a grid is formed over the space of possible robot base configurations, the robot is placed at each of the grid points to compute collision-free IKS . Those base configurations that yield at least one IKS for each of the task point will be labeled as feasible. For each feasible base configuration, the shortest collision-free path that visits all task points is computed using the VP algorithm described in the previous section. Finally, the robot base configuration with the shortest path length is selected as the optimum.

4 Examples

We have used our algorithm for a robot engaged in a simulated spot welding task. A CRS-465 robot shown in Figure 2 is to perform 4 spot welds on a car body at the points marked with circles. The two filled circles are to be welded from under the roof while the two empty circles are to be welded from above. The robot is placed at a nominal position and its joint angles are set to the home position as shown in Figure 2. A simplified L-shaped weld gun is used to avoid clutter in showing motions. When computing collision-free paths, we used a simplied model of the robot shown in Figure 3 to speed up the computation of distances between the robot and obstacles. We note here that fast distance calculation routines for complex objects have been developed in [1, 11] and will be integrated in our algorithm in the near future. The CRS-465 robot can have up to 8 inverse kinematic solutions for a given tool configuration. We used only half of the solutions since the other half amounts to rotating the robot shoulder 180 degrees and swinging the first and second links of the arm to the other side. (Such motions take a long time to execute and are not used in practice.) Because the spot weld process is symmetric with respect to the roll about the axis of the welding-gun tip, we generated inverse kinematic solutions at a 15 degree roll interval. The number of collision-free inverse kinematic solutions for each spot-weld point ranges from 2 to 8. The robot base is moved in (x, y, θ) space on a grid of 5x7x5 points. The computation time to examine 175 robot base locations was 50 hours on an SGI Indigo 2 workstation with codes that include graphics. The time should be reduced at least by a factor of 3 when running without the graphics. The shortest path length corresponding to each base configuration is shown in Table 1. The optimal base configuration has an offset of $(0.0, -10.0, -5.0)$ from the nominal base configuration shown in Figure 2. Figures 4(a) through 4(d) show the robot base configuration minimizing the length of the spot-welding path, the inverse kinematic solutions used at the weld points, and the order of spot welding. A supercomputer or a network of workstations will be needed to obtain a similar run time for a realistic problem containing complex objects and many more weld points.

5 Conclusions

We have developed an algorithm that determines the optimal base configuration for robots engaged in visit-point tasks. This algorithm provides a systematic approach to guarantee the optimality of the base configuration and to compute the actual motion of the robot performing the tasks. This algorithm significantly reduces the robot cycle time as well as the labor costs associated with determining the base configuration and the robot motion visiting the points. Ultimately, the use of this algorithm reduces the time from design to mass production, giving manufacturers a competitive edge.

Our algorithm can be applied to many other robot planning applications including curve-tracing tasks (seam welding and caulking) and cover-surface tasks (painting). The current optimization criterion is the minimum path length in the joint space, but other measures such as maximum force, dexterity and especially

the cycle time as done in [4] can be considered. Minimizing cycle time prefers a path consisting of a smaller number of monotonic path segments over a geometrically shorter path with many short segments. When computing resources are limited, our algorithm can be run first on a coarse grid of the base configurations to identify the neighborhoods of the optimal base configuration, followed by a fine grid search to exactly locate the optimal configuration.

References

[1] Canny, J.F. and Lin, M., "A Fast Algorithm for Incremental Distance Computation," *Proc. of IEEE Int. Conf. on Robotics and Automation,* pp. 1008-1014, Sacramento, CA, 1991.

[2] Chen, P.C. and Hwang, Y.K., "SANDROS: A Motion Planner with Performance Proportional to Task Difficulty," *Proc. of IEEE Int. Conf. on Robotics and Automation,* pp. 2346-2353, Nice, France, 1992.

[3] Cook, J.S. and Han, B. T., "Optimal Robot Selection and Work Station Assignment for CIM System," *IEEE Trans. on Robotics and Automation,* vol. 10, no. 2, pp. 210-219, April 1994.

[4] Feddema, J.T., "Kinematically Optimal Robot Placement for Minimum Time Coordinated Motion," *Proc. IEEE Int. Conf. on Robotics and Automation,* Minneapolis, MN, 1996.

[5] Hwang, Y.K. 1994, "New Regimes for Robot Motion Planning," Proc. of RI/SME Fifth World Conference on Robotics Research, pp. 2/13-23 ,Cambridge, MA.

[6] Hwang, Y.K. and Ahuja, N., "Gross Motion Planning - A Survey," *ACM Computing Surveys* vol 24, no 3, pp. 219-292, September 1992.

[7] Hwang, Y.K., Chen, P.C., Neidigk, D.D. and Maciejewski, A.A. 1994, "A Global Motion Planner for Curve-Tracing Robots," Proc. of 1994 IEEE International Conference on Robotics and Automation, pp. 662-667, San Diego, CA.

[8] Latombe, J.C., *Robot Motion Planning*, New York: Kluwer Academic Publishers, 1991.

[9] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B., *The Traveling Salesman Problem*, New York: John Wiley & Sons, 1985.

[10] Robgoderer, U. and Woenckhaus, C., "A Concept for Automatical Layout Generation," *Proc. of IEEE Int. Conf. on Robotics and Automation,* pp. 800-805, Nagoya, Japan, 1995.

[11] Xavier, P.G., "A Generic Algorithm for Constructing Hierarchical Representations of Geometric Object," *Proc. IEEE Int. Conf. on Robotics and Automation,* Minneapolis, MN, 1996.

[12] *IGRIP User's Manual*, Deneb Robotics, Inc., 1995.

[13] *CimStation User's Manual*, Silma, Inc., 1995.

Table 1: Tour length verses Robot base configuration. The '-' denotes that one of the weld points is not reachable. The 'f' denotes that the point-to-point planner cannot find a collision-free path either because there is no solution or because it could not find one in 100 second time limit. The minimum path is achieved at the base configuration of (0,-10,-5).

y \ x	θ	-10	-5	0	+5	+10
-20	-10	-	-	-	-	-
	-5	-	-	-	-	-
	0	-	-	-	-	-
	+5	-	-	-	-	-
	+10	-	-	-	-	-
-15	-10	-	-	-	-	-
	-5	-	-	-	-	-
	0	-	-	-	-	-
	+5	-	-	-	-	-
	+10	-	-	-	-	-
-10	-10	21.20	-	-	-	-
	-5	20.04	20.02	20.00	20.29	22.13
	0	f	20.32	20.28	f	f
	+5	20.45	20.40	20.34	f	f
	+10	20.62	21.60	21.05	21.04	22.01
-5	-10	21.16	f	f	f	f
	-5	20.08	20.05	20.78	20.38	21.55
	0	f	f	f	f	26.71
	+5	21.21	f	f	f	f
	+10	20.67	22.14	21.22	20.84	21.50
0	-10	31.06	f	f	f	f
	-5	20.85	21.39	20.88	20.43	21.61
	0	f	f	f	f	f
	+5	21.35	f	f	f	f
	+10	20.87	20.65	20.96	20.79	21.50
+5	-10	-	f	f	f	f
	-5	-	20.98	20.69	20.52	21.37
	0	-	21.94	f	f	f
	+5	-	f	f	f	f
	+10	-	f	20.54	20.36	21.81
+10	-10	f	f	f	f	f
	-5	28.57	29.36	30.15	32.39	32.34
	0	22.17	f	21.72	f	f
	+5	f	f	f	f	f
	+10	20.90	f	20.59	21.94	21.87

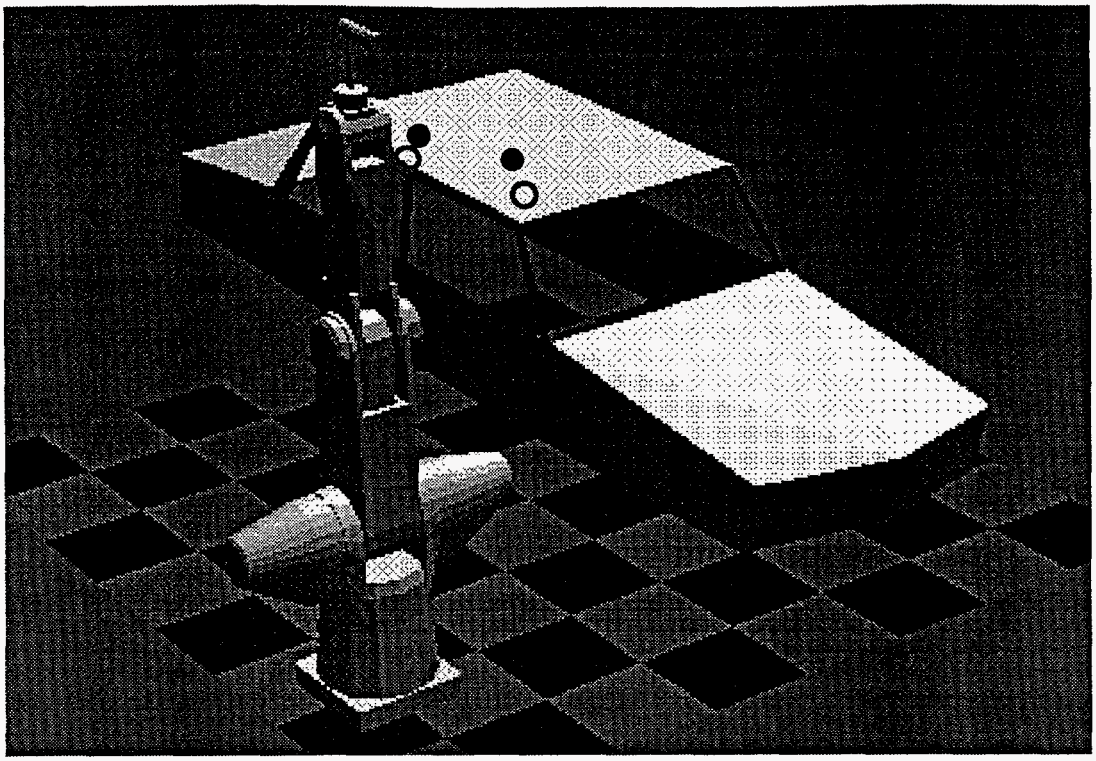

Figure 2. Planning a CRS–465 robot to spot weld a car body. The weld points are shown as circles on the roof. The filled circles are to be welded from under the roof, while the empty circles are to be welded from above the roof.

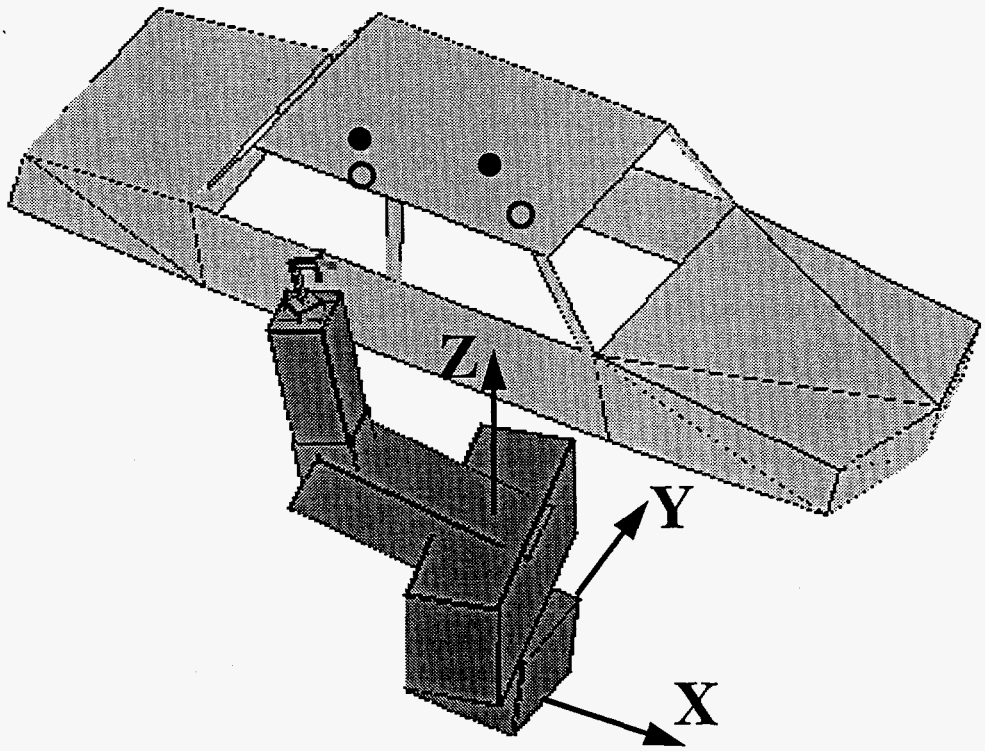

Figure 3. Simplified models of the robot links are used to plan collision–free motions. The robot base is varied in x, y, θ to find the optimal base configuration for the spot welding task. The ranges are ([–10, 10], [–20, 10], [–10, 10]), and the step sizes are (5,5,5), respectively. For reference, the forearm of the robot is 330 units long.

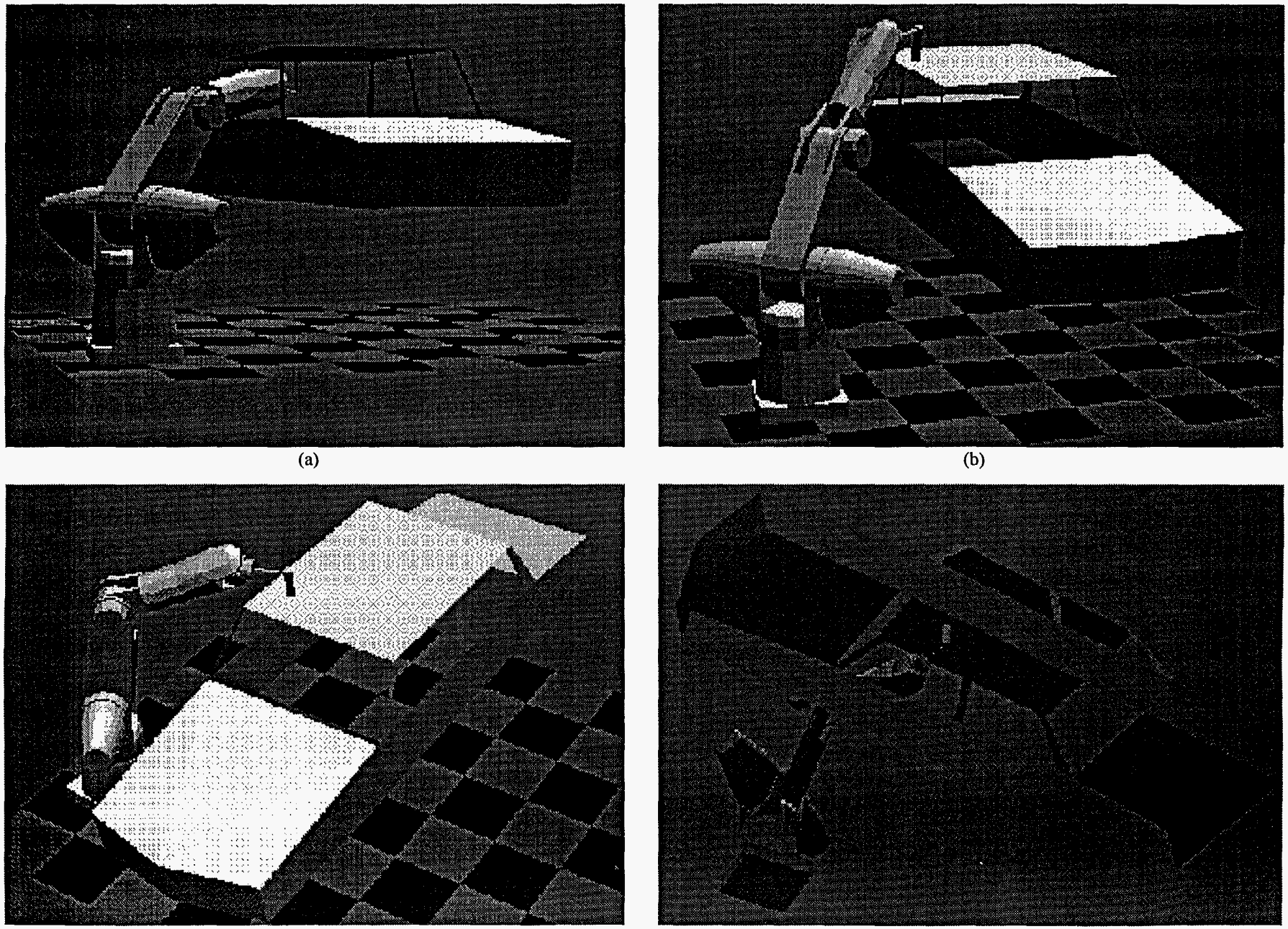

Figure 4. The optimal base configuration has a (0.0, –10.0, –5) offset from the nominal configuration in Figure 2. Figures (a) through (d) show the robot configuration used for each weld point as well as the order of spot welding.