Null Space Optimization for Effective Coverage of 3D Surfaces using Redundant Manipulators

Jürgen Hess Gian Diego Tipaldi Wolfram Burgard

***Abstract*— In this paper we consider the problem of null space minimization in coverage path planning of 3D surfaces for redundant manipulators. Existing coverage solutions only focus on Euclidean cost functions and often return suboptimal paths with respect to the joint space. In the approach described here, we explicitly consider the null space by treating different inverse kinematics solutions as individual nodes in a graph and model the problem as a generalized traveling salesman problem (GTSP). The GTSP is a generalization of the TSP where the nodes of the graph are subdivided into clusters and at least one node in each cluster needs to be visited. We evaluate our approach using a PR2 robot and complex objects. Our results demonstrate that our method outperforms Euclidean coverage algorithms in terms of manipulation effort and completion time.**

I. Introduction

Coverage of 3D surfaces is becoming an important and interesting problem for personal robotics, mainly due to its interesting and potential applications (e.g., autonomous cleaning, painting, or scraping of complex 3D objects). As for today, cleaning services are envisioned to be one of the most relevant applications of mobile service robots in the near future. Prassler and Kosuge [12] list thirteen commercially available domestic cleaning robots, all of which are floor cleaning robots and thus operate on a 2D planar environment. To the best of our knowledge there is no manipulation robot that can clean arbitrary 3D surfaces.

In this paper we consider the problem of coverage path planning for robotic manipulators where the task space is constrained to lie on the surface and specific costs in joint space need to be minimized. We further assume that the orientation of the end effector is orthogonal to the direction of travel, which is the case for a variety of different tools including paintbrushes, sponges, and squeegees.

A popular solution to the problem is to transform the surface into a graph and solve the associated traveling salesman problem (TSP). Although this approach is well suited in order to minimize the Euclidean path length, it limits the possibility to define appropriate cost functions in joint space to perform null space optimization. There are two reasons for this limitation. The first reason is that the orientation of the end effector is only known after a full path is available. The second reason is that the cost of travel from a node to another one depends on the configuration of the robot in the first node. This configuration is not unique for redundant manipulators and may depend on the sequence in which the nodes are visited.

All authors are with the Autonomous Intelligent Systems Lab, Computer Science Department, University of Freiburg, Germany. {hess,tipaldi,burgard}@informatik.uni-freiburg.de

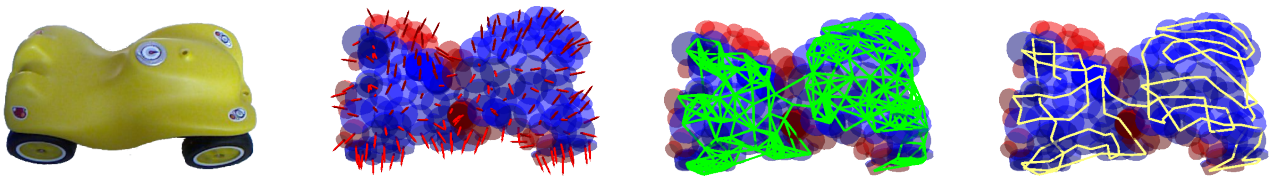
Fig. 1. PR2 robot using a sponge to clean a bobby car.

To overcome these limitations, we model the problem as a generalized traveling salesman problem (GTSP), a generalization of the TSP, where a set of clusters is defined over the nodes. As a result, our approach generates coverage strategies that are optimized with respect to user-defined cost functions over the joint-space.

We evaluate our approach using real data collected from a PR2 robot. Fig. 1 shows a typical setup of our experiments, where the robot is confronted with the task of cleaning the surface of a bobby car using its arm. The results show that our approach generates paths covering the object while minimizing the target cost defined by the above-mentioned cost functions. We evaluate our approach using two different cost functions, which are the completion time and the distance in the joint space. According to the results, our method outperforms Euclidean coverage algorithms with respect to both cost functions.

II. Related Work

In general, the term *coverage path planning* refers to the problem of finding a path in a fully connected graph that covers all nodes and minimizes some cost measure. Most of the approaches for coverage assume that the environment is known and seek the shortest path that traverses each location once, which corresponds to the traveling salesman problem (TSP). Finding the optimal solution for a TSP is well-known to be NP-hard. Practical solutions for 2D surfaces typically rely on heuristics to reduce the problem size or



(a) Object Image (b) Surface Patches and Normals (c) Euclidean Graph (d) Joint-GTSP Solution

Fig. 2. Illustration of the coverage planning process. A solution generated with the Joint-GTSP is shown in (d).

utilize specific structures of environments.

Gabriely et al. [7], for example, decompose the surface into a grid and suggest different coverage strategies based on spanning trees. Other approaches use a decomposition into non-overlapping cells of different shapes. Latombe, for example, uses a trapezoidal decomposition [10]. Another approach is the Boustrophedon cellular decomposition [5] which divides the free space into cells which can be covered with vertical back and forth motions that can be connected across the cells. Huang et al. [9] use this decomposition and compute an optimal coverage path by minimizing the number of turns of the robot. Mannadiar and Rekleitis [11] propose a graph structure based on the Boustrophedon cellular decomposition and show that a complete minimal path through this graph can be computed in polynomial time. All of these approaches, however, assume a planar robot moving on a 2D plane.

Recently, coverage algorithms have also been extended to non-planar surfaces. Xu et al. [14], for example, extend the work of Mannadiar and Rekleitis [11] to the field of aerial coverage of terrain with unmanned aerial vehicles (UAVs). Cheng et al. [4] focus on 3D urban coverage with UAVs. Coverage has also been addressed in the field of spray painting automotive parts. Atkar et al. [1] show how simple automotive parts like convex bent sheets can be covered such that the resultant paint deposition on the target surface achieves acceptable uniformity. In these applications however, the robot does not operate on the surface. They also do not address the problem of minimizing costs in the configuration space of the robot and do not consider robotic manipulators. Breitenmoser et al. [3] extended 2D coverage for mobile robots to 3D surfaces, by using Voronoi tessellations to map the surface to a 2D plane. Although closely related, they only considered mobile robots moving on the surface and coverage in terms of Euclidean distance.

This paper presents a novel solution to the problem of covering 3D surfaces with a redundant manipulator. In contrast to the majority of previous work, our method furthermore addresses the problem of null space minimization.

III. Formulating a Coverage Problem as a GTSP

In this section we will present the formulation of the coverage problem in terms of a GTSP. The GTSP is a generalization of the TSP, where a set of clusters is defined over the nodes. Each solution to a GTSP includes at least one node from each cluster and, as in the TSP, the goal is to find a tour with minimum cost. In our case, we are confronted with a special version of the GTSP where the clusters do not intersect and where we search for a tour that visits each cluster exactly once.

First, we show how we convert the surface of the object into set of locally planar patches. Then, we describe how we use this representation to generate a Euclidean graph which encodes collision-free traveling paths over the surface. Finally, we use this graph to construct a GTSP that solves the coverage problem by minimizing a user-defined cost function in joint space.

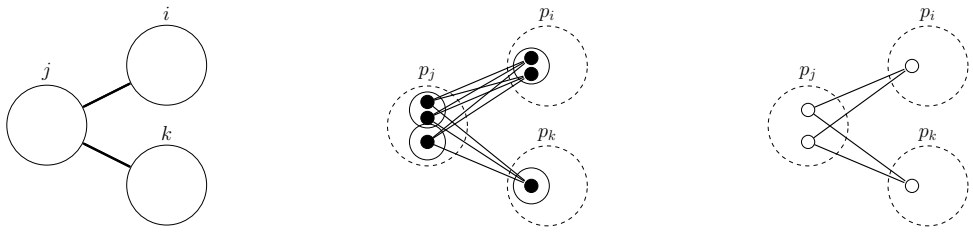
A. Euclidean Graph Construction from Point Clouds

Our robot is equipped with a Kinect sensor that generates a point cloud of the object to be covered. We approximate the resulting point cloud with a set of planar patches, which comprises the object model. Given the point cloud, we first randomly select a point. We then determine the points that lie within an ε-neighborhood, where the value of ε depends on the size of the tool used to accomplish the task, and use RANSAC to fit a plane. We accept the plane as a new surface patch if the root mean square error (RMSE) is below a threshold and mark the points used for calculation. The process is restarted with the remaining points until all points are marked or were drawn. An example of the resulting surface representation is shown in Fig. 2b.

Having modeled the surface as a set of locally planar patches and their surface normals, we aim at constructing a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, with the following properties:

- The nodes correspond to the set of reachable patches.
- The edges represent collision free paths for the end effector.

The construction of the graph proceeds as follows. We check reachability of each patch. If a patch is not reachable, we mark it accordingly and delete the node from the graph. For each remaining node, we select the top k nearest neighbors within a radius r and connect them linearly with an edge. We then simulate an end effector movement along each of the edges and check them for collision. If there is no valid path along an edge, we delete the edge from the graph. As a result, each node can be reached on a collision free path along the edges through the graph. If the graph construction results in more than one connected component, we apply the optimization approach described below to each of the components seperately. The resulting graph is shown in Fig. 2c. The red patches were marked as not reachable.



(a) TSP (b) Joint-GTSP (c) Curvature-GTSP

Fig. 3. The different graph representations used. Edges between nodes respectively clusters which are not adjacent in the Euclidean graph are assigned a constant high weight and not visualized in the figure.

B. GTSP for Joint Space Minimization

To perform joint space minimization, we formulate the problem as a GTSP which we obtain by extending the Euclidean graph to account for the possible joint configurations for each pose. Each cluster p_i of the GTSP corresponds to a node i in the Euclidean graph. For each edge $e_{i,j}$ in the Euclidean graph, we sample a set of inverse kinematics solutions for both the start position i and the end one j. The inverse kinematics solutions are samples in the null space and due to redundancies. The solution sets are then inserted in the respective clusters and used in the GTSP. Nodes within the same cluster are not connected. Fig. 3b illustrates the resulting graph. The nodes corresponding to joint space solutions are marked solid black. The solid circles mark the ends of each edge in the Euclidean graph and the dotted circles the clusters.

The final step is to define the cost functions of the GTSP in terms of joint space configurations. In this paper we are interested in two cost functions, namely the manipulation effort and the time to completion. For simplicity we model the manipulator as being able to be controlled in velocity and neglect the dynamics and accelerations. Note that they can be easily taken into account by appropriately modifying the expressions to be computed. We define the distance between two nodes with respect to the manipulation effort as the total amount of displacement of each joint between two configurations:

$$dist(p_i^k, p_j^l) = \sum_m \|\Delta q_m\|, \tag{1}$$

where p_i^k and p_j^l are the node k and l of the clusters p_i and p_j and Δq_m is the displacement of the m-th joint between the respective joint configurations. Similarly, with respect to the time to completion, the distance is the minimal time needed for the movement between the joint configurations assuming maximum velocity $v_{\max}$:

$$dist(p_i^k, p_j^l) = \max_m \left(\frac{\|\Delta q_m\|}{v_{\max}}\right). \tag{2}$$

The cost assigned to edges in the GTSP graph are given by:

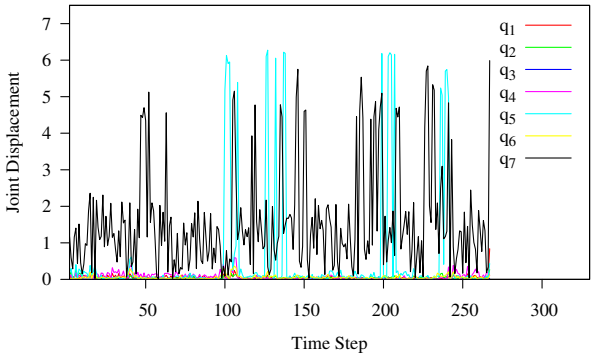
$$c(p_i^k, p_j^l) = \begin{cases} dist(p_i^k, p_j^l), & \text{if } A(i,j) = 1 \text{ and } i \neq j \\ h, & \text{if } A(i,j) = 0 \text{ and } i \neq j \\ \infty, & \text{if } i = j \end{cases}, \tag{3}$$

where A is the adjacency matrix of the Euclidean graph and $A(i,j) = 1$ if the two nodes i and j are adjacent. We assign a constant value h to all edges leading from p_i^k to p_j^l if the nodes i and j are not connected in the Euclidean graph. This ensures that a solution can be found as the clusters are fully connected. The quantity h is set to a high value to indicate that those edges require additional path planning. Having constructed this graph, we transform the GTSP into a TSP using the method of Behzad et al. [2]. We then compute the solution for the TSP using dedicated solvers described below. As a result we obtain an effective coverage path with respect to the cost function selected. In the following, we refer to this GTSP as Joint-GTSP.

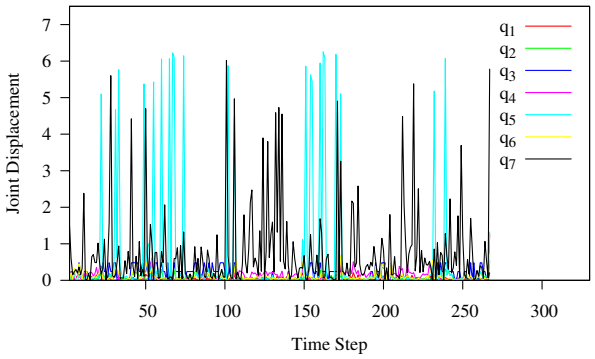
IV. Hierarchical Approximation for Efficient Planning

Unfortunately, the size of the GTSP described above is exponential in the number of nodes that correspond to inverse kinematics solutions, which limits us to only a few inverse kinematics samples. In this section we describe a hierarchical approximation of the Joint-GTSP that scales quadratically in the number of inverse kinematics solutions and exponentially only in the number of edges in the Euclidean graph.

To find a suitable approximation, we analyzed the cost profile of each joint in both the Joint-GTSP and the TSP solution (see Fig. 4). The TSP solution was obtained by using the Euclidean graph for solving for a solution, computing the end effector orientations along the path and querying for inverse kinematics solutions. Fig. 4 shows that most of the gain of the GTSP solution is due to a cost reduction of the joint corresponding to the end effector orientation. Minimizing the effort of this joint in turn means minimizing the curvature of the path in Euclidean space. Using this insight, we decided to decouple the full minimization problem by first optimizing for the end effector orientation and then optimizing the remaining joints. More formally, we first generate a simplified GTSP problem that minimizes a weighted cost function on the Euclidean distance and the curvature of the path on the manifold (Curvature-GTSP). As in the general case, each cluster p_i of the Curvature-GTSP corresponds to node i in the Euclidean graph. We then consider only each start and end point of an edge in the Euclidean graph as a node in the Curvature-GTSP. Thus, the number of nodes in a cluster p_i only corresponds to the number of edges of node i in the Euclidean graph and is



(a) TSP



(b) Joint-GTSP

Fig. 4. Joint displacement for different TSP solutions. Joint q_7 corresponds to the end effector. Note the reduced impact of joint q_7 when using the Joint-GTSP.

independent of the number of inverse kinematics samples. Each node p_i^k in the graph defines its own coordinate system, where the x axis is oriented with respect to the direction of travel and the z axis with respect to the normal of the patch. Let $\Delta X = \begin{bmatrix} t & R(u,\theta) \end{bmatrix}$ be the transformation between the coordinate frames of the nodes k and l of the clusters p_i and p_j, where t is the translational and $R(u,\theta)$ the rotational part expressed in axis-angle notation. The distance between the two nodes is then:

$$dist(p_i^k, p_j^l) = (1-\beta)\,\|t\| + \beta\,\theta, \tag{4}$$

where β is a parameter, weighting between the Euclidean distance and the curvature of the path. The cost assigned to edges in the Curvature-GTSP graph are then computed in the same way as in the general setting described in Eq. 3.

Fig. 3c shows an example of such a graph. The dotted circles denote the nodes in the Euclidean graph and the clusters in the GTSP. The smaller circles denote the end points of each edge in the Euclidean graph and form the new set of nodes in the Curvature-GTSP.

As the second step we solve the Curvature-GTSP using the same reduction to a TSP as for the general case. For further optimization of the path in joint space, we construct a directed source to target graph of joint space positions (see Fig. 5). The graph is constructed in the following way. From the TSP solution we extract the sequence of 6 DoF end

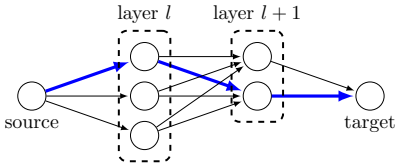
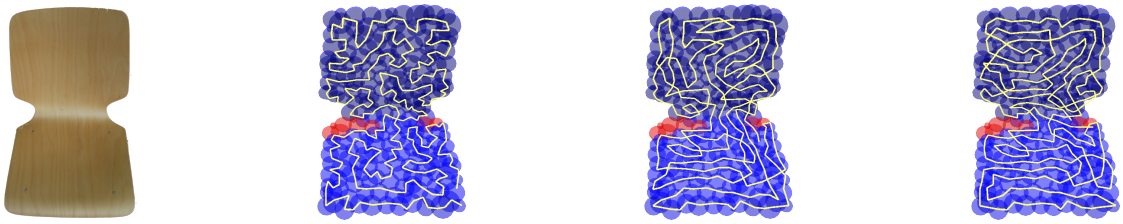


Fig. 5. Directed source to target graph. Each node corresponds to a joint configuration and each layer to one end effector pose. The thick (blue) path marks the solution.

effector positions of the tour. For each position, we compute a set of inverse kinematics solutions that forms the nodes of an intermediate layer l of the graph. Thus, each layer corresponds to one end effector pose from the path and each node to one inverse kinematics solution. We select the current joint position of the manipulator as the source and connect it to the first layer. The first layer is determined as the position of the TSP tour closest to the current end effector position. All nodes in layer l are connected forward to all nodes in layer $l+1$. The target node of the graph is an artificial sink as we do not require an exact final joint position. The weight of each edge in this graph is equivalent to the cost functions computed between the joint configurations of the respective nodes in the Joint-GTSP. The solution is then found using the Dijkstra algorithm to compute the shortest path through the graph which is visualized in Fig. 5.

V. Experiments

We validated our approach using real data recorded with the PR2 mobile manipulation robot. The data has been recorded from two different objects, a chair and a bobby car, using a Kinect sensor. We used a single view of each object but our approach also extends to multiple views that are fused, for example using the method of Ruhnke et al. [13]. We chose the bobby car and the chair for their very different surface structure. The bobby car is a complex non-convex object whereas the chair is largely planar. For the construction of the Euclidean graph we chose the number of nearest neighbors k to be 8 and set the search distance to 10 cm to limit the graph complexity. These settings resulted in a sufficiently dense graph. The β parameter in the cost function of the Curvature-GTSP (see Eq. (3)) was set to 0.9, favoring solutions with smoother curvature changes. The exact setting was not crucial for the experiments. We evaluated three approaches, the Joint-GTSP, the Curvature-GTSP, and the TSP with a different number of inverse kinematics samples. For the TSP construction, we used the Euclidean graph and also fully connected the graph, setting a constant high weight to all edges not adjacent in the Euclidean graph. After the tour construction, we computed the end effector orientation and optimized the joint space in the same way as for the Curvature-GTSP. For the cases with a small number of samples < 10, we manually sampled the joint around the middle of the configuration space. The 100 samples were obtained by sampling uniformly. We also simulated a non-redundant manipulator by fixing one of the joints. This is equivalent to the one sample case described below. The



(a) Chair (b) TSP (c) Curvature-GTSP (d) Joint-GTSP

Fig. 6. Sample coverage paths for the chair experiment.

TABLE I

RESULTS FOR THE CHAIR EXPERIMENT, OPTIMIZING FOR TASK TIME (LEFT) AND EFFORT (RIGHT).

	Task Time			**Effort**		
	TSP	Curvature-GTSP	Joint-GTSP	TSP	Curvature-GTSP	Joint-GTSP
	1 Sample			1 Sample		
Dist. [m]	9.54 ± 0.00	12.03 ± 0.23	12.26 ± 0.18	9.54 ± 0.00	12.11 ± 0.25	12.09 ± 0.21
Effort	336.46 ± 3.34	228.25 ± 5.21	218.59 ± 5.05	335.88 ± 3.27	228.70 ± 5.67	**215.65 ± 3.37**
Task Time [s]	153.46 ± 1.55	97.76 ± 2.17	**94.58 ± 2.15**	153.19 ± 1.51	98.15 ± 2.34	95.43 ± 1.56
Calc. Time [s]	3.84 ± 0.36	114.40 ± 9.20	118.03 ± 6.68	4.12 ± 0.48	133.27 ± 8.07	131.06 ± 7.83
	3 Samples			3 Samples		
Dist. [m]	9.54 ± 0.00	12.04 ± 0.12	13.03 ± 0.26	9.54 ± 0.00	12.19 ± 0.19	13.10 ± 0.35
Effort	327.66 ± 2.29	220.41 ± 2.97	214.24 ± 5.69	305.39 ± 1.96	204.59 ± 3.28	**183.73 ± 4.02**
Task Time [s]	138.34 ± 1.47	87.15 ± 1.48	**84.53 ± 2.64**	141.67 ± 1.42	93.48 ± 1.97	86.98 ± 1.99
Calc. Time [s]	3.73 ± 0.15	129.60 ± 7.76	1649.11 ± 150.90	4.19 ± 0.31	137.88 ± 8.50	1808.6 ± 182.45
	100 Samples			100 Samples		
Dist. [m]	9.54 ± 0.00	12.09 ± 0.16	N/A	9.54 ± 0.00	12.18 ± 0.30	N/A
Effort	303.54 ± 4.93	205.17 ± 4.53	N/A	206.81 ± 1.44	**134.49 ± 5.66**	N/A
Task Time [s]	101.60 ± 1.53	**69.91 ± 1.47**	N/A	159.46 ± 3.57	101.56 ± 1.51	N/A
Calc. Time [s]	23.92 ± 0.69	148.42 ± 8.64	N/A	12.01 ± 0.41	152.72 ± 14.40	N/A

samples were obtained by fixing one joint and then using OpenRAVE [6] to calculate the inverse kinematics solution. The TSP has been solved using the LKH solver, a state-of-the-art TSP solver based on the Lin-Kernighan heuristic [8]. Due to a random element in the selection of the initial tour of the LKH solver, we repeated the experiments 10 times.

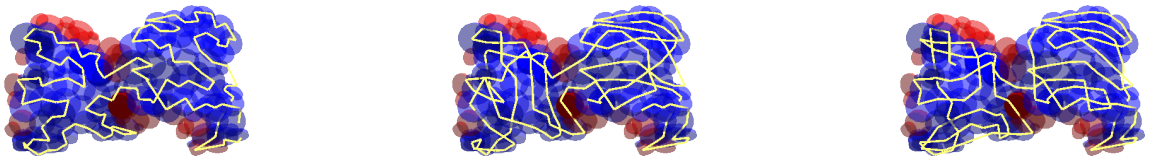
The results of our experiments are shown in Table I for the chair and in Table II for the bobby car. To illustrate the difference of the Cartesian path on the surface of the objects, a sample solution for both objects can be found in Fig. 6 and Fig. 7. For both objects and minimization strategies, i.e., effort and time (see Eq. (1) and Eq. (2)), we compute the total Euclidean distance, the total effort, and the total time for task completion as well as the calculation time. In the calculation of the task completion time, we assume a maximum velocity of $v_{\max} = 2\,\text{rad/s}$ for each joint and a bang-bang velocity profile which is equivalent to impulsive accelerations. The calculation time specifies the time needed for solving for a coverage path given the Euclidean graph.

The table shows the results for one sample (no redundancy), the maximum number of samples usable for the Joint-GTSP (three for the chair and nine for the bobby car), and 100 samples (only for the TSP and the Curvature-GTSP). As can be seen, the TSP results in the shortest Cartesian path but also in the highest effort and execution time. This comes with no surprise as it is not possible to encode these costs in the Euclidean graph. More interestingly, this also shows that the shortest Euclidean path is not always the best one with respect to execution time. The Joint-GTSP results in a significant reduction in terms of effort and time although the length of its Cartesian path increases.

For both the one sample and the maximum samples cases, we see that the Joint-GTSP and the Curvature-TSP perform significantly better than the TSP in both experiments (minimum effort and minimum time) while the Joint-GTSP performs slightly better. If we increase the number of samples to 100, we see that the Curvature-GTSP is able to perform better than the Joint-GTSP at the maximum number of usable samples, with no significant overhead on the calculation time.

VI. CONCLUSION

In this paper we presented a novel approach to perform null space minimization for coverage path planning problem on 3D surfaces. Existing coverage algorithms mostly focus on robots moving on a planar surface, minimizing Euclidean properties on the plane. We showed that when considering redundant manipulators this property does not hold anymore and costs in joint space need to be explicitly considered. We showed how these costs can be expressed by modeling the problem in terms of a generalized traveling salesman problem and presented a general framework for null space



(a) TSP (b) Curvature-GTSP (c) Joint-GTSP

Fig. 7. Sample coverage paths for the bobby car experiment.

TABLE II

RESULTS FOR THE BOBBY CAR EXPERIMENT, OPTIMIZING FOR TASK TIME (LEFT) AND EFFORT (RIGHT).

	Task Time			**Effort**		
	TSP	Curvature-GTSP	Joint-GTSP	TSP	Curvature-GTSP	Joint-GTSP
		1 Sample			1 Sample	
Dist. [m]	5.89 ± 0.00	7.61 ± 0.19	7.48± 0.15	5.89 ± 0	7.49 ± 0.12	7.53± 0.16
Effort	206.66± 3.52	180.27 ± 3.52	157.29 ± 4.02	205.33 ± 3.49	176.95 ± 6.95	**153.07 ± 3.57**
Task Time [s]	85.00 ± 0.36	72.08 ± 1.65	**63.31 ± 1.59**	84.80 ± 0.60	70.92 ± 2.45	63.67 ± 1.48
Calc. Time [s]	1.06 ± 0.25	30.01 ± 1.89	31.07± 3.38	1.11 ± 0.24	32.16 ± 2.51	26.58 ± 0.81
		9 Samples			9 Samples	
Dist. [m]	5.89 ± 0.00	7.88 ± 0.22	8.49 ± 0.29	5.89 ± 0.00	7.73 ± 0.14	8.97 ± 0.30
Effort	202.70 ± 3.93	174.92 ± 7.90	169.28 ± 6.13	171.67 ± 2.22	143.83 ± 4.15	**137.50 ± 7.78**
Task Time [s]	72.25 ± 0.56	61.60 ± 1.93	**60.85 ± 1.65**	81.00 ± 0.41	67.10 ± 2.55	68.42 ± 3.37
Calc. Time [s]	1.15 ± 0.09	30.84 ± 1.86	1549.44 ± 183.41	1.17 ± 0.13	31.98 ± 2.10	2255.48 ± 213.98
		100 Samples			100 Samples	
Dist. [m]	5.89 ± 0.00	7.60 ± 0.15	N/A	5.89 ± 0.00	7.61 ± 0.16	N/A
Effort	202.19 ± 3.84	164.03 ± 4.04	N/A	159.70 ± 2.67	**131.88 ± 5.86**	N/A
Task Time [s]	68.14 ± 0.32	**54.86 ± 1.34**	N/A	85.44 ± 0.90	68.63 ± 3.05	N/A
Calc. Time [s]	4.29 ± 0.29	35.25 ± 3.43	N/A	2.47 ± 0.36	33.50 ± 1.94	N/A

optimization of arbitrary cost functions. We further showed an efficient approximation of the general approach that scales quadratically with the number of inverse kinematics samples. The approach has been evaluated using real data collected from a PR2 robot. Results obtained with real-world objects show that we are able to obtain paths that cover the object thereby minimizing a user-defined cost function in the null space. They furthermore show that our approach outperforms Euclidean coverage algorithms in terms of manipulation effort and completion time. Interestingly, the experiments also show that the shortest path in Euclidean distance is not always the best one with respect to execution time. In the future we plan to extend the approach to incorporate the mobility of the base and the potentially more flexible bi-manual manipulation.

REFERENCES

[1] P.N. Atkar, A. Greenfield, D.C. Conner, H. Choset, and A.A. Rizzi. Uniform coverage of automotive surface patches. *The International Journal of Robotics Research*, 24(11):883–898, 2005.

[2] A. Behzad and M. Modarres. A new efficient transformation of the generalized traveling salesman problem into traveling salesman problem. In *Proc. of the Intl. Conf. of Systems Engineering*, pages 6–8, 2002.

[3] A. Breitenmoser, J. Metzger, R. Siegwart, and D. Rus. Distributed coverage control on surfaces in 3d space. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5569–5576, 2010.

[4] P. Cheng, J. Keller, and V. Kumar. Time-optimal UAV trajectory planning for 3d urban structure coverage. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2750–2757, 2008.

[5] H. Choset and P. Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Intl. Conf. on Field and Service Robotics*, 1997.

[6] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, 2010.

[7] Y. Gabriely and E. Rimon. Spanning-tree based coverage of contin uous areas by a mobile robot. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, volume 2, pages 1927–1933, 2001.

[8] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.

[9] W.H. Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, volume 1, pages 27–32, 2006.

[10] J.C. Latombe. *Robot motion planning*. Springer Verlag, 1990.

[11] R. Mannadiar and I. Rekleitis. Optimal coverage of a known arbitrary environment. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 5525–5530, 2010.

[12] E. Prassler and K. Kosuge. Domestic robotics. In *Springer Handbook of Robotics*, pages 1253–1281. Springer, 2008.

[13] M. Ruhnke, R. Kümmerle, G. Grisetti, and W. Burgard. Highly accurate 3d surface models by sparse surface adjustment. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2012.

[14] A. Xu, C. Viriyasuthee, and I. Rekleitis. Optimal complete terrain coverage using an unmanned aerial vehicle. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2513–2519, 2011.