Planning human-aware motions using a sampling-based costmap planner

Jim Mainprice[1,2], E. Akin Sisbot[1,2], Léonard Jaillet[3], Juan Cortés[1,2], Rachid Alami[1,2], Thierry Siméon[1,2]

{jmainpri, sisbot, jcortes, rachid, nic}@laas.fr, ljaillet@iri.upc.edu

[1]CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France

[2]Université de Toulouse; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France

[3]Institut de Robótica i Informática Industrial ; CSIC-UPC ; c/ Llorens i Artigas 4-6, 08028 Barcelona, Spain

***Abstract*— This paper addresses the motion planning problem while considering Human-Robot Interaction (HRI) constraints. The proposed planner generates collision-free paths that are acceptable and legible to the human. The method extends our previous work on human-aware path planning to cluttered environments. A randomized cost-based exploration method provides an initial path that is relevant with respect to HRI and workspace constraints. The quality of the path is further improved with a local path-optimization method. Simulation results on mobile manipulators in the presence of humans demonstrate the overall efficacy of the approach.**

I. INTRODUCTION

In an environment where robots and humans co-exist and work together, robot motions need to explicitly take into account the presence of humans. Therefore, hardware as well as software components need to be designed by considering human's safety [5], [15]. Besides ensuring safety in robot hardware with compliant designs [20], [1], the motions of the robot need to be planned in a "human-aware" manner.

In previous work [17], [18], we have presented a motion planner that explicitly takes into account human-robot constraints (e.g. their relative distance, the human's field of view and posture) to synthesize navigation and manipulation motions. This planner was based on human-robot user studies [11], as well as on existing human-human space sharing theories [8]. The proposed method was to our knowledge the first to investigate a "planning" approach to the problem of human-robot intelligent space sharing. HRI constraints were represented through cost functions depending respectively on the human kinematic model, field of view and accessibility. This representation of the problem led to costmaps defined over the workspace. Motion planning was solved using grid search techniques for planning object motions, and generalized inverse kinematics for the robot to follow the planned object path. While this decoupled approach is sufficient in the absence of strong workspace constraints, it may fail in cluttered environments such as shown in Figure 1.

In this paper we extend the capabilities of the planner using sampling-based planning algorithms, which enable planning in the robot configuration space and finding human-aware motions in cluttered environments. Sampling-based path planning methods [2], [13], are able to handle complex problems in high-dimensional spaces. However, they usually operate in a binary configuration space, aiming to find feasible collision-free solutions rather than optimal paths. Moreover, due to their probabilistic nature, solution paths have generally low quality, and a post-processing phase is commonly used to improve them locally regarding specific criteria (e.g. length, clearance). The proposed method relies on the recent algorithm T-RRT [9], which computes good-quality paths given a general cost-function defined over the robot configuration space. The solutions provided by T-RRT are further improved using local optimization methods also described in the paper. Finally, we present a refined description of the HRI constraints, in particular the "arm comfort constraint" only used in [18] for computing the object transfer point of hand over tasks while considered here for planning robot motions.

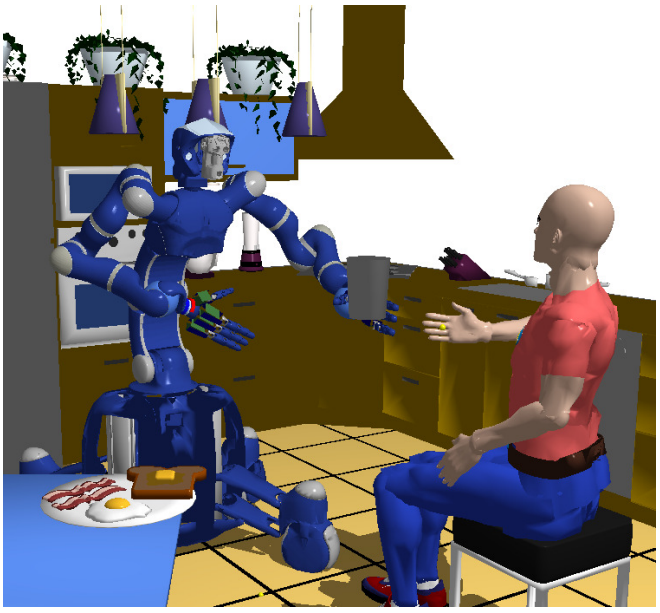
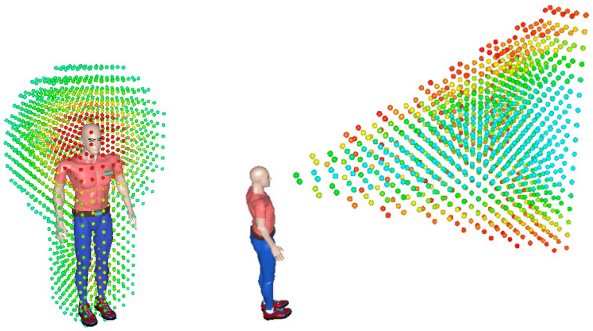


Fig. 1. A cluttered environment such as a home provides a difficult workspace for robot motion planning in which the human presence adds new constraints. In this paper we propose to use a sampling-based method to achieve high-dimensional optimal planning regarding cost functions designed to take explicitly the human into account.

The paper is organized as follows. Next section describes the model of the HRI constraints. Section III presents the path planning method. First, the T-RRT algorithm, which is used to find a first good-quality path, is briefly explained.



(a) Distance costmap (b) Visibility costmap

Fig. 2. The costmaps model the distance and visibility constraint by assigning to each point of the cartesian space an HRI cost. The safety cost function is inversely proportional to its distance to the human while the visibility cost function reasons about the field of view modeled by the gaze direction.

Then, we describe post-processing methods that can be applied to further improve path quality over a configuration-space costmap. Finally, experimental results are presented to demonstrate the efficacy of the approach (Section IV).

II. Human-Robot Interactions constraints

The presence of humans in a robot workspace brings new constraints to navigation and manipulation planning. In this work, several examples of important constraints have been taken into account, such as safety, visibility and arm comfort which are further detailed. These constraints have to be considered as examples of the broad variety of HRI properties that can be taken as input of our planner.

The first constraint depicted in Figure 2, called distance constraint, mainly focuses on ensuring the safety of the interaction by controlling the distance between the robot and the human. Only an approximate bounding volume of the human body without considering the arm geometry is used for the distance computation. This safety constraint, which is reasonable, given that the focus is set on preventing any risk of harmful collision between the human and the robot, keeps the robot away from the head and body. Moreover, it has been also shown in proxemics theory [8] that violation of an intimate space radius generates a feeling of intrusion. Therefore the farther a point is situated from the human, the lesser its HRI safety cost is, until some maximum threshold at which it becomes null.

The second constraint, called visibility constraint, has the purpose of limiting the human's surprise as the robot is moving in the workspace. A human will feel less surprise if the robot stays in sight resulting in a safer and more comfortable interaction as shown in [17]. Thus each workspace point has a cost proportional to the angle between the gaze and its position in Cartesian space as illustrated in Figure 2.

The third constraint, called arm comfort constraint, was introduced in [18] to compute object transfer position in hand over tasks with the human. This section presents a refined description of this constraint, that is also considered by the motion planner in order to generate paths for which it is easy for the human to access an object held by the robot at any time. For this the robot must reason on humans' accessibility and kinematics. The presupposed human reaching volume can be preprocessed using generalized inverse kinematics (GIK). For each position inside the reaching volume, the torso configuration is determined to stay as close as possible to a given resting posture. Collision detection against the environment is used to further validate those postures. Then, to each valid reaching posture is assigned a comfort cost as shown in Figure 3 by using the predictive human-posture cost function introduced in [14]. The comfort is estimated by the sum of the three functions:

- The first function computes a joint angle distance from a resting posture q^N to the actual posture where q is the configuration of the human:

$$f_1 = \sum_{i=1}^{DOF} w_i (q_i - q_i^N)^2$$

- The second considers the potential energy of the arm which is defined by the difference of the arm and forearm heights with those of a resting posture (Δz_i) pondered by an estimation of the arm and forearm mass m_i :

$$f_2 = \sum_{i=1}^{2} (m_i g)^2 (\Delta z_i)^2$$

- The third penalizes configuration close to joint limits. To each joint corresponds a minimum and a maximum limit and the distance to the closest limit (Δq_i) is taken into account in the cost function as follows:

$$f_3 = \sum_{i=1}^{DOF} \gamma_i \Delta q_i^2$$

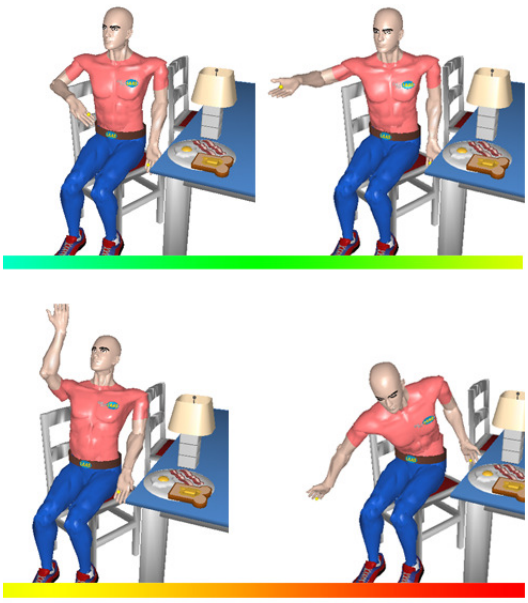


Fig. 3. Arm comfort: Four poses that vary from comfortable and natural on the upper left corner to uncomfortable and uneasy postures on the lower right corner, the color gradient expresses the corresponding cost function value.

Each constraint expressed as an elementary three-dimensional costmap is combined with a weighted sum as follows:

$$c(h,x) = \sum_{i=1}^{N} w_i c_i(h,x),$$

where h is the human model posture and x the point of the workspace for which the cost is computed. In the current implementation the weights are defined manually and the cost functions are evaluated "on the fly" during planning .

The planners of [17], [18] were based on a direct search on the resulting cartesian grids to produce legible and comfortable motions using basic graph search techniques. In [17], navigation tasks were performed based on 2D cost grids explored by an A* algorithm. Extension to manipulation tasks [18] led to cartesian grids used to compute the end effector path, assuming the computed path was feasible for the robot. While such a decoupled approach is sufficient in the absence of strong environmental constraints, it may fail in situations where the path planned for the object can not be followed by the robot because of collisions with workspace obstacles. Hence, the extension to costmaps defined over the robot configuration space is desirable.

III. Planning Low Cost Paths

Instead of considering a grid on the workspace, a cost is defined for each robot configuration as:

$$c(h,q) = \sum_{i=1}^{N} w_i c_i(h, FK(q)),$$

where q is a configuration and FK the robot's forward kinematics function. Given the resulting configuration-space costmap, we adopt a sampling-based algorithm for computing good-quality paths. Several approaches have been introduced in former works, in particular RRT variants [6], [19], [3] in the context of field robotics or more recently in [10]. In this work, we apply a more general algorithm, called T-RRT [9], briefly explained below. This section also presents a new algorithm for local optimization of the solution through a post-processing phase that also handles such general cost functions.

A. Costmap exploration

The T-RRT algorithm [9] takes advantage of the performance of two methods. First, it benefits from the exploratory strength of RRT-like planners resulting from their expansion bias toward large Voronoi regions of the space. Additionally, it integrates features of stochastic optimization methods, which apply transition tests to accept or reject potential states. It makes the search follow valleys and saddle points of the cost-space in order to compute low-cost solution paths (see Figure 4). Several criteria can be used to measure the quality of a path based on its parametric cost function (e.g. the maximal cost, the average cost, the integral cost along the path, or the mechanical work). The T-RRT algorithm aims at finding paths that minimize the mechanical work criterion,

Algorithm 1: Transition-based RRT

```
input  : the configuration space CS;
         the cost function c : CS → R+*;
         the root q_init and the goal q_goal;
output : the tree T;
begin
    T ← InitTree(q_init);
    while not StopCondition(T, q_goal) do
        q_rand ← SampleConf(CS) ;
        q_near ← NearestNeighbor(q_rand, T);
        q_new ← Extend(T, q_rand, q_near);
        if q_new ≠ NULL
        and TransitionTest(c(q_near), c(q_new), d_near−new)
        and MinExpandControl(T, q_near, q_rand) then
            AddNewNode(T, q_new);
            AddNewEdge(T, q_near, q_new);
end
```

but also simultaneously satisfy other quality metrics such as the integral cost.

Algorithm 1 shows the pseudo-code of the T-RRT planner. Similarly to the Extend version of the basic RRT algorithm [12], a configuration q_{rand} is randomly sampled. It yields both the nearest tree node q_{near} to be extended, and the extension direction. The extension from q_{near} is performed toward q_{rand} with an increment step δ, which has to be small enough to avoid missing cost picks in the presence of binary obstacles. Thus, if the new portion of the path leads to a collision, a null configuration is returned and the extension fails independently of the associated costs. This extension process ensures the bias toward unexplored free regions of the space. The goal of the second stage is to filter irrelevant configurations regarding the search of low cost paths before inserting q_{new} in the tree.

Such filtering is performed by the `TransitionTest` function in which the probability of acceptance of a new configuration is defined by comparing its cost c_j relatively to the cost c_i of its parent configuration in the tree. It relies on the Metropolis criterion, commonly used in stochastic optimization methods, with a transition probability p_{ij} used to penalize cost-increasing motions and defined as follows:

$$p_{ij} = \begin{cases} exp(-\frac{\Delta c^*_{ij}}{K \cdot T}) & \text{if } \Delta c^*_{ij} > 0 \\ 1 & \text{otherwise.} \end{cases} \quad (1)$$

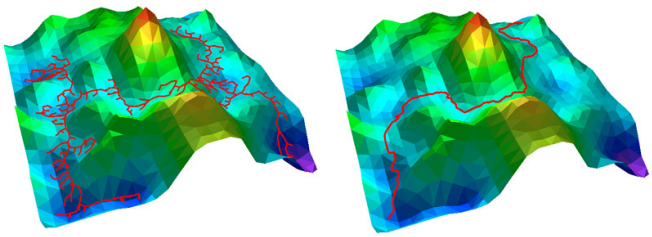


Fig. 4. T-RRT constructed on a 2D costmap (left). The transition test favors the exploration of low-cost regions, resulting in good-quality paths (right).

Algorithm 2: Random Cost Shortcut

```
input  : The Path P;
output : The Path P;
begin
    while not StopCondition() do
        (q1, q2) ← P.getTwoConfig();
        LP ←getSegment(q1, q2) ;
        if isValidAndLowerCost(LP, q1, q2) then
            P.ReplacePortion(LP, q1, q2) ;
end
```

Algorithm 3: Random Path Perturbation

```
input  : The Path P;
output : The Path P;
begin
    while not StopCondition() do
        q_perturb ← P.shootRandConfigOnPath();
        (q_near1, q_near2) ← P.getNeigh (q_perturb, step);
        q_rand ← shootRandDirection() ;
        q_new ← Expand(q_perturb, q_rand, step) ;
        LP1 ←getSegment (q_near1, q_new) ;
        LP2 ←getSegment (q_new, q_near2) ;
        LP ← LP1 + LP2 ;
        if isValidAndLowerCost(LP, q_near1, q_near2)
        then
            P.ReplacePortion (LP, q_near1, q_near2, );
end
```

This test integrates a self-tuning method in order to automatically control its filtering strength, and thus ensures a minimal growth of the tree. Finally, the `MinExpandControl` function forces the planner to maintain a minimal rate of expansion toward undiscovered regions of the space. Thus, it avoids possible blocking situations during the search. For more details on T-RRT refer to [9].

B. Smoothing

Traditional post-processing methods (e.g. [7]) are generally applied to reduce the length and/or to increase the clearance of a path. We investigate below the extension of path optimization techniques to the case of a general cost function. First, we present how the classic shortcut method can be extended, and secondly, we present a new method that optimizes locally the path by random perturbations.

1) Random shortcut method: The cost-space extension of the shortcut method is similar to the original approach, but the cost of the path is tested together with collisions and kinematic constraints. The method is sketched in Algorithm 2. At each iteration, the new path portion replaces the current one only if it is feasible and of lower cost.

This method reduces the length of the input path while improving its quality, and usually converges rapidly to a local minimum. However when applied to more general cost functions to be optimized, the fact that the optimization is restricted inside the convex hull defined by the set of points on the path appears to be a limiting factor. Hence, it is not sufficient to deform the path towards the low-cost valleys of the costmap.

2) Random path perturbation: The goal of this method is to avoid the path convex hull limitation. The path is iteratively deformed by moving a point $q_{perturb}$ randomly selected on the path in a direction determined by a random sample q_{rand}. This process (depicted in Figure 5) creates a deviation from the current path, The new segment replaces the current segment if it has a lower cost. Collision checking and kinematic constraints verification can be performed before or after cost comparison depending on the computational cost of both processes.

The first step of this method (sketched in Algorithm 3) is to select a configuration $q_{perturb}$. This selection is biased to higher cost segments. For this, path portions are sorted according to their cost, and high-cost portions have higher chances of being picked for optimization. In a second phase an extension toward a random direction q_{rand} is performed to select a configuration q_{new} inside the path's neighborhood.

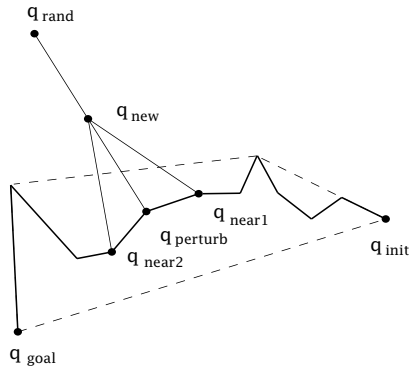


Fig. 5. The random path perturbation does not restrict the new path to be inside the convex hull defined by the path's set of points enabling a more global exploration of the input path neighborhood.

The $step$ parameter [1] controls the amplitude of the local perturbations. First it is used to determine the distance between the two configuration named q_{near1} and q_{near2} that are selected to be both at $step/2$ of $q_{perturb}$. It also controls the deviation of q_{new} from the current path, chosen as a percentage of the $step$ parameter. High values of the $step$ parameter tend to deform strongly the path while low values tend to refine locally the solution.

This perturbation method explores more globally the path neighborhood, but usually outputs longer paths. Therefore It is complementary to the shortcut method that tends to shorten the path. Thus, in the post-processing phase, it may be suitable to use a combination of the two methods in order to obtain smooth and low-cost solutions. Also note that these methods can be used to optimize an input path resulting from the RRT search that does not take cost into account. This is illustrated in the results presented below

[1] The $step$ parameter can be set at a certain percentage of the path length, in the experimental results, we use 10% which refines locally the path in a convenient way. The distance between q_{new} and $q_{perturb}$ is set to be at 25% of $step$.

that also compare the computational performance and path quality obtained with global T-RRT costmap planning and local path-optimization of RRT solutions.

IV. Experimental results

The T-RRT algorithms and the path smoothing methods have been implemented into the path planning software *Move3D* [16]. The experiments reported below have been performed on a 2.6GHz INTEL processor. All performance results summarized in the tables correspond to average values computed over 10 runs.

The scenario illustrates a planning problem involving torso and arm motions of the mobile *Justin* robot from DLR [4]. *Justin* hands an object to the human in a kitchen environment cluttered by ceiling lamps. The resulting motion planning problem involves 10 active DoFs that enable the robot to bend the torso while moving the arm in order to avoid the ceiling lamps. Taking into account the HRI constraints induces high-dimensional costmaps to be explored for generating safe and legible robot motions.

In Figure 6, the three costmaps are taken separately into account during motion planning resulting in three different solutions to the same problem. Figure 6(c) illustrates the effect of the distance criterion on the robot motion. As one can see, the resulting path pushes the robot farther from the human and causes a safer behavior compared to the direct path generated by the "standard planner". Similarly in Figure 6(d), with visibility cost function, the robot moves the object in a way that it stays as visible as possible to the human. Finally considering the reachability criterion in Figure 6(e), the robot moves the object on a path that maximizes the possibility for the human to reach the object in a comfortable way.

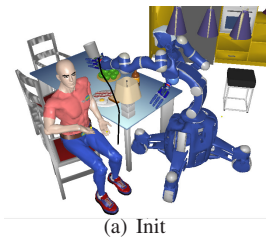
The total planning time including post-processing and the integral cost of solutions before and after optimization are compared on those three costmaps in Table I. The exploration phase using either a standard RRT or T-RRT are compared. The reported times include a $4sec$ post-processing performed by iteratively interleaving runs of the $perturbation$ and $shortcut$ methods introduced in section III.

T-RRT planning on the first costmap improves drastically the cost of the solutions because the solution path with low integral cost implies a large detour from the standard RRT solution. As one can see on Figure 6(d) and 6(e) low cost object translation paths on the visibility costmap

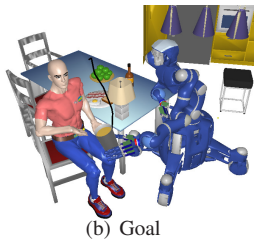
TABLE I

RUNS ON AN ELEMENTARY COSTMAP

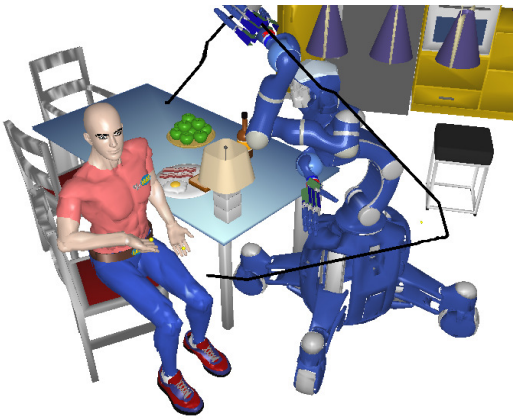
	Time(sec.)	Cost(Before)	Cost(After)
Dist.			
RRT	8.57	212	111
T-RRT	13.31	45	18
Visib.			
RRT	8.64	294	186
T-RRT	7.05	251	176
Reach.			
RRT	8.39	158	89
T-RRT	15.91	117	62



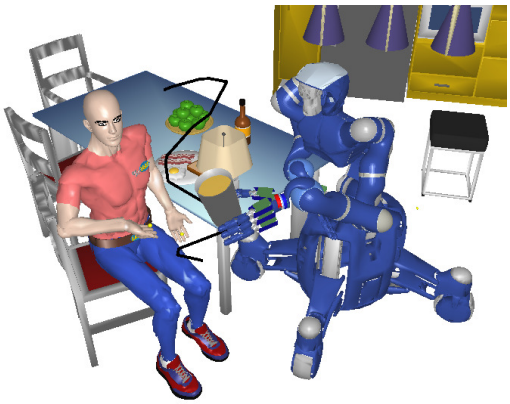
(a) Init



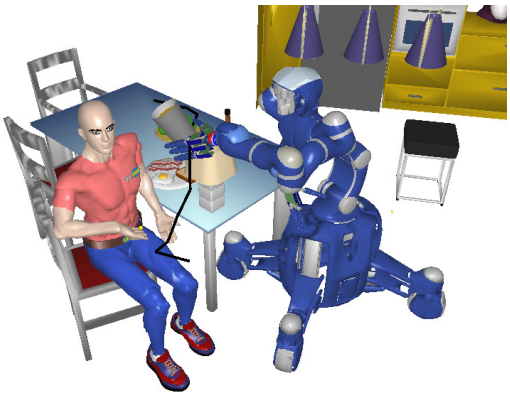
(b) Goal



(c) Distance Cost

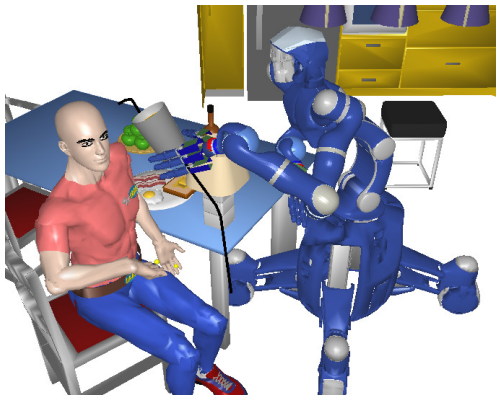


(d) Visbility Cost

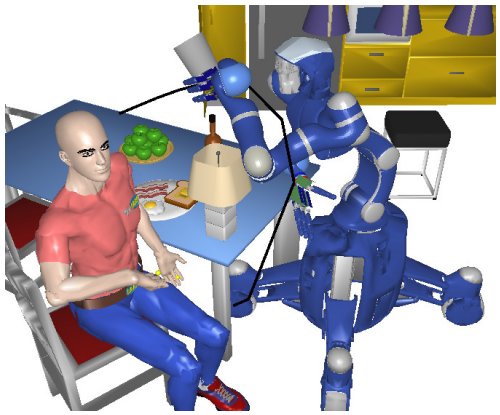


(e) Reachability Cost

Fig. 6. Justin robot in a handover task. The workspace is cluttered by object on the table and ceiling lamps which make a difficult motion planning problem. The three cost functions lead to three cost-spaces which can be taken separately as input of the proposed sampling-based costmap planner.



(a) RRT path



(b) Costmap path

Fig. 7. A complete object hand over example scenario. The robot has the object in its hand. The human is sitting on a chair looking away from the robot. While the motion planned with a standard planner does not consider the presence of the human, the one planned by taking into account the three constraints generates a comfortable motion. By following this path the robot stays as visible as possible, as sufficiently far as possible and the object is comfortable to reach by the human.

TABLE II

RUNS ON THE COMBINATION OF THE 3 CONSTRAINTS

	Time(sec.)	Cost(Before)	Cost(After)
RRT	8.65	179	120
T-RRT	16.69	82	54

and reachability constmap are geometrically closer to the standard RRT path of Figure 7(a). This explains the less significant gain found of using the cost-space planning in Table I. Finally, Table II similarly reports the results obtained when considering the combination of the three costmaps the solution obtained realizes a good compromise between the three constraints illustrated on Figure 7.

V. CONCLUSION

We have presented a novel cost-space planning approach for computing human-aware motions considering HRI constraints. The proposed T-RRT algorithm bridges the gap between low dimensional costmap planning and sampling-based motion planning. We have also described a path post-processing technique that can be used to further improve T-RRT solutions, or for the local optimization of paths computed without considering the HRI constraints.

Future work concerns the extension of the proposed framework to more general HRI constraints and better characterizing the desired properties of human-aware motions. Currently the planner considers a static human model that does not account for possible human motions during the execution of a the robot path. We expect to enhance our models with a new framework to overcome this limitation. We also plan to further validate the approach through experiments with real robot systems.

VI. ACKNOWLEDGMENTS

This work has been supported by the European Community's Seventh Framework Program FP7/2007-2013 "DEXMART" under grant agreement no. 216239.

REFERENCES

[1] A. Bicchi and G. Tonietti, *Fast and soft arm tactics: Dealing with the safety-performance trade-off in robot arms design and control*, Robotics and Automation Magazine (2004).

[2] H.M. Choset, S. Hutchinson, K.M. Lynch, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation*, The MIT Press, 2005.

[3] R. Diankov and J. Kuffner, *Randomized statistical path planning*, IEEE/RSJ Int. Conf. on Intel. Rob. And Sys., 2007.

[4] C. Ott et al., *A humanoid two-arm system for dexterous manipulation*, IEEE-RAS Int. Conf. on Hum. Robot., 2006.

[5] R. Alami et al., *Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges*, Procceedings IROS Workshop on pHRI (Beijing, China), 2006.

[6] A. Ettlin and H. Bleuler, *Randomised Rough-Terrain Robot Motion Planning*, IEEE/RSJ Int. Conf. on Intel. Rob. And Sys., 2006.

[7] R. Geraerts and M.H. Overmars, *Creating high-quality paths for motion planning*, The International Journal of Robotics Research (2007).

[8] Edward T. Hall, *A system for the notation of proxemic behavior*, American anthropologist (1963).

[9] L. Jaillet, J. Cortés, and T. Siméon, *Sampling-based path planning on configuration-space costmaps*, IEEE Transactions on Robotics (2010).

[10] S. Karaman and E. Frazzoli, *Incremental Sampling-based Algorithms for Optimal Motion Planning*, Robotics: Science and Systems (RSS) Conference, 2010.

[11] K. L. Koay, E. Akin Sisbot, D. A. Syrdal, M. L. Walters, K. Dautenhahn, and R. Alami, *Exploratory study of a robot approaching a person in the context of handling over an object*, Association for the Advancement of Artificial Intelligence Spring Symposia, AAAI (Palo Alto, CA, USA), March 2007.

[12] S. LaValle and J. Kuffner, *Rapidly-exploring random trees: Progress and prospects*, Workshop on the Algorithmic Foundations of Robotics, AK Peters, Ltd., 2001.

[13] S.M. LaValle, *Planning algorithms*, Cambridge Univ Pr, 2006.

[14] R.T. Marler, S. Rahmatalla, M. Shanahan, and K. Abdel-Malek, *A new discomfort function for optimization-based posture prediction*, (2005).

[15] S. Nonaka, K. Inoue, T. Arai, and Y. Mae, *Evaluation of human sense of security for coexisting robots using virtual reality. 1st report: evaluation of pick and place motion of humanoid robots*, IEEE Int. Conf. Robot. And Autom. (New Orleans, USA), April 2004.

[16] T. Simeon, JP Laumond, and F. Lamiraux, *Move3D: a generic platform for path planning*, 4th Int. Symp. on Assembly and Task Planning, Citeseer, 2001.

[17] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Siméon, *Human aware mobile robot motion planner*, IEEE Transactions on Robotics (2007).

[18] E. A. Sisbot, L. F. Marin Urias, R. Alami, and T. Siméon, *Spatial reasoning for human-robot interaction*, IEEE/RSJ Int. Conf. on Intel. Rob. And Sys. (San Diego, CA, USA), November 2007.

[19] C. Urmson and R. Simmons, *Approaches for heuristically biasing RRT growth*, 2003.

[20] M. Zinn, O. Khatib, B. Roth, and J. K. Salisbury, *Playing it safe [human-friendly robots]*, IEEE Robotics & Automation Magazine (2004).