Multi-Objective Cost-to-Go Functions on Robot Navigation in Dynamic Environments

Gonzalo Ferrer[1] and Alberto Sanfeliu[1]

***Abstract*— In our previous work [1] we introduced the Anticipative Kinodynamic Planning (AKP): a robot navigation algorithm in dynamic urban environments that seeks to minimize its disruption to nearby pedestrians. In the present paper, we maintain all the advantages of the AKP, and we overcome the previous limitations by presenting novel contributions to our approach. Firstly, we present a multi-objective cost function to consider different and independent criteria and a well-posed procedure to build a joint cost function in order to select the best path. Then, we improve the construction of the planner tree by introducing a cost-to-go function that will be shown to outperform a classical Euclidean distance approach. In order to achieve real time calculations, we have used a steering heuristic that dramatically speeds up the process. Plenty of simulations and real experiments have been carried out to demonstrate the success of the AKP.**

I. Introduction

A great interest has grown lately in the deployment of service robots in urban environments. However, the complexity of these scenarios is notable, since there exist multiple interactions between pedestrians, static obstacles, and robots.

In this paper, we discuss in depth the importance of the different cost functions in navigation algorithms and we present a method to systematically reduce the complexity associated to consider multiple criteria while performing robot navigation, such as avoiding collisions, minimize the impact to pedestrians, reach the goal efficiently, *etc*.

In our previous work [1] we presented a novel navigation algorithm, the Anticipative Kinodynamic Planner (AKP), that calculates the reaction produced by its planned trajectory and provides the minimum impact to nearby pedestrians. Unfortunately, there were some limitations that have motivated the present work: the navigation algorithm was highly conditioned by the learning environment used, while we desire a more general approach for different kinds of scenarios.

To alleviate this problem, we present a multi-objective function that considers different and independent objectives, and a technique to correctly handle and compare these different objectives into a single and well-posed function.

We propose a cost-to-go function, based on the multi-objective costs to calculate distances between states, instead of a Euclidean-based distance. The cost-to-go metric demonstrates to greatly enhance the area covered by the set of paths calculated by the planner.

[1]The authors are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC. Llorens Artigas 4-6, 08028 Barcelona, Spain. {gferrer,sanfeliu}@iri.upc.edu.

This work was supported by the Spanish Ministry of Science and Innovation project DPI2013-42458-P.

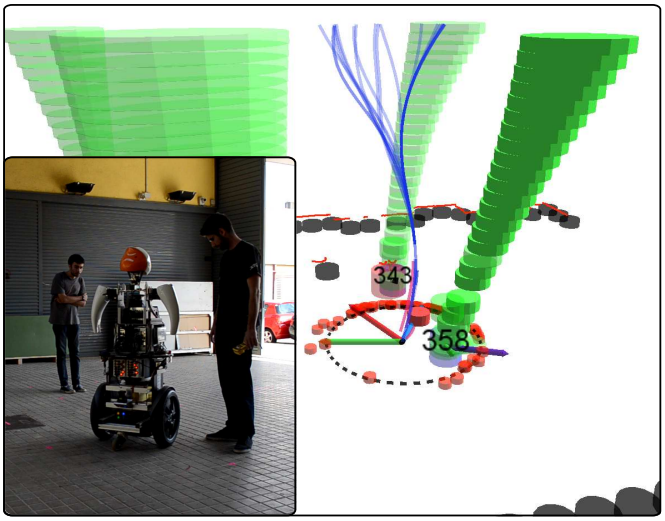

Fig. 1. Anticipating kinodynamic planning performing in a real scenario. The robot goal (red cylinder) is between two people.

In order to speed up the calculations, we use a steering heuristic to connect states, and at the same time, we introduce a randomness factor to the steering function, to obtain more distinct paths. The Social Force Model (SFM) [2], and the Extended SFM [3] have been used in the present work as the steering method.

As can be seen in Fig. 1, the AKP predicts people's positions (green cylinders) with respect to time (z axis) and calculates a set of paths (blue lines), taking into account time restrictions, since dynamical obstacles change their positions over time. Among the set of calculated paths, we chose the minimum cost path.

II. Related work

Finding a cost function to correctly characterize robot navigation among people is not an easy endeavor, and may become an ill-posed problem, *e.g.* local minima, *etc*. In [4], the authors have addressed this problem and proposed a set of homotopically distinct trajectories. Sharing this same goal, we tackle this problem by proposing a multi-objective cost function where we optimize different independent criteria. A Multi-Objective Optimization (MOO) method [5], [6] has inspired a solution to the minimization problem.

A joint calculation of people's path and the robot's path has been proposed in [7] using Gaussian processes and a distance-based *interaction potential* between people. This method is able to provide anticipatively, with respect to the scene, the less occupied robot's trajectory, but unlike our approach, we are able to quantify the magnitude of the alterations to nearby pedestrians and plan accordingly.

The prediction method used in the present work is a geometrical-based predictor [8], [9] that infer human motion intentions to subsequently predict human motion in a continuous space.

Some state of the art approaches in robot navigation [10], [11], [12] use a wide variety of techniques such as potential fields, sampling, *etc*. We follow a sampling-based technique to solve the planning problem. In general, sampling methods [13], [14] can take into account kinematic and dynamic constraints, which is of great relevance in dynamical environments. However, the distance function used can change completely the outcome of the planner. Several works [15], [16], [17] make use of cost-to-go functions as a distance function between states. We propose a cost-to-go function based on the same multi-objective cost function mentioned above and calculate distances between states for the RRT planning framework.

Modern optimal approaches [18], [19] provide a collision free path for multiple dynamic obstacles. Steering methods are used in state of the art approaches, such as feedback control [20] or analytical solutions through relaxation of constraints [17]. The objective is the same, reducing the computational cost drastically. Although it may seem appealing, the price to pay may be a reduction of the search space. We propose a technique to minimize this problem by introducing some randomness into the steering function.

III. Anticipating kinodynamic planning

Before presenting the planning contributions, we will briefly describe the Extended Social Force Model (ESFM) [3], based on [2], since it provides a realistic model describing interactions among humans in typical social environments [9].

The ESFM considers humans and robots as free particles in a 2D space abiding the laws of Newtonian mechanics, and is based on attractors and repulsors. The resultant force is defined as

$$\boldsymbol{f}_n = \boldsymbol{f}_n^{goal}(\mathcal{D}_n) + \sum_{j \in P \setminus n} \boldsymbol{f}_{n,j}^{int} + \sum_{o \in O} \boldsymbol{f}_{n,o}^{int} + \sum_{r \in R} \boldsymbol{f}_{n,r}^{int}, \quad (1)$$

where interactions with people, obstacles and robots are considered for the nth person. We will use (1) in the planning algorithm and it is presented in more detail in [3].

The output of the ESFM are forces that describe human trajectories. We can calculate the corresponding accelerations and the differential constraint

$$\dot{s}_z = dc(s_z, u_z) \quad (2)$$

that describes the propagation of the state s_z, depending on z being a robot or a person.

In our previous work [1] we presented the basic structure of the AKP (Alg. 1), which produces a set of path candidates in an RRT fashion. The main features can be summarized as:

- A kinodynamic solution is calculated satisfying time restrictions due to pedestrians up to $t_{horizon} = t + h$.
- Prior requirement: the calculation of a valid global path.

Algorithm 1 AKP($q_r^{goal}, s_{ini}, t_{horizon}, K$)

```
Initialize T(V, E) ← {∅}
V ← s_ini
{q_pi^goal} = PEOPLE_INTENTIONALITY()
for j = 1 to K do
    [q_r^g, β] = SAMPLE(q_r^goal)
    s_parent = FIND_NEAREST_VERTEX(q_r^g, T, β)
    [U^new, S^new] = EXTEND(s_parent, q_r^g, {q_pi^goal})
    J^new = COST_TO_GO(U^new, S^new, q_r^goal, T)
    if NO_COLLISION(S_new) then
        V ← V ∪ {S^new, J^new}
        E ← E ∪ {U^new}
    end if
end for
return MINIMUM_COST_BRANCH(T)
```

- Anticipative planning: prediction information is dependent on the robot path calculated by considering the joint state $s = [s_r, s_{p_1}, \ldots, s_{p_N}]$ consisting of the robot and all he pedestrians.
- At each iteration, the AKP provides a locally valid set of commands $U = \{u(t), \ldots, u(t+h)\}$, and only the first command is executed.
- The path computed minimizes the perturbations on the scene, according to a cost function.

Despite the contributions and the overall novelty of the work, there were important limitations that have motivated the present paper. The cost function defined as a weighted-sum of heterogeneous terms was not able to correctly handle the cost of the robot navigation and its impact towards the environment in multiple scenarios. We learned the parameters for certain configurations, which in general did not hold to all scenarios.

In the present work, we maintain all the advantages of the previous AKP, and we overcome the previous limitations by developing and integrating the following novel contributions to our approach:

1) A multi-objective cost function to consider different and independent criteria in a well-posed procedure.
2) A cost-to-go function to measure the cost to connect a pair of states, instead of a Euclidean distance.
3) Introducing randomness into the steering method to sample more effectively the solution space.

As stated before, our main objective is to navigate in urban environments where there are dynamical obstacles (pedestrians). We use the basic mechanics of the kinodynamic RRT [13] to extend paths, in order to generate a large set of feasible paths and chose the best path according to multiple minimization criteria. Some of the proposed contributions aim to obtain more distinct paths and thereby improve the search for a better trajectory, since we dispose of a more representative sample of the solution space. In Sec. VI is described the improvement of our contributions

IV. RRT EXTEND: ESFM STEERING HEURISTIC

The ESFM serves for the prediction algorithm used by our approach, and as an inspiration for the steering heuristic in order to connect the pair of poses q_{ini} and q_f, in a computationally fast manner.

We calculate using (1) the resultant robot force $\boldsymbol{f}_r$, which takes into account its environment $[s_{p_1}, \ldots, s_{p_N}]$, while at the same time tries to reach the given random goal q_r^g.

The EXTEND function is depicted in Alg. 2, where it firstly propagates the robot state $s_r(s_{parent})$ according to u_r^{new} and integrates the differential equation (2) by using numerical integration. Then, for every person on the scene, and if the person has not reached its inferred goal $q_{p_i}^{goal}$ (line 5 in Alg. 2), an action $u_{p_i}^{new}$ is calculated depending on the rest of the dynamical obstacles on the scene and the new robot state s_r^{new} (line 6 in Alg. 2) in a cooperative way.

Algorithm 2 EXTEND$(s_{parent}, q_r^g, \{q_{p_i}^{goal}\}, \beta)$

1: **while** $t(s_{parent}) < t_{horizon}$ & $s_r(s_{parent}) \not\subseteq q_r^g$ **do**
2: $\quad u_r^{new} = \boldsymbol{f}_r(s_{parent}, q_r^g)/m_r$
3: $\quad s_r^{new} = s_r(s_{parent}) + \int_{\Delta t} \dot{s}_r\big(s_{parent}, u_r^{new}\big)$
4: $\quad$ **for** $i = 1, \ldots, N$ **do**
5: $\quad\quad$ **if** $s_{p_i}(s_{parent}) \not\subseteq q_{p_i}^{goal}$ **then**
6: $\quad\quad\quad u_{p_i}^{new} = \boldsymbol{f}\big(q_{p_i}^{goal}, s_{parent}, s_r^{new}\big)/m_i$
7: $\quad\quad\quad s_{p_i}^{new} = s_{p_i}(s_{parent}) + \int_{\Delta t} \dot{s}_{p_i}\big(s_{prnt}, u_{p_i}^{new}\big)$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: $\quad s_{parent} = [s_r^{new}, s_{p_1}^{new}, \ldots, s_{p_N}^{new}]$
11: $\quad S^{new} \leftarrow S^{new} \cup s_{parent}$
12: $\quad U^{new} \leftarrow U^{new} \cup [u_r^{new}, u_{p_1}^{new}, \ldots, u_{p_N}^{new}]$
13: **end while**
14: **return** $[S^{new}, U^{new}]$

The EXTEND function propagates the state of the system up to $t_{horizon}$ or until the robot reaches the random goal q_r^g.

A. Randomness in the steering function

Steering methods present excellent characteristics to drastically reduce the computational cost of kinodynamic planners, both in the EXTEND function as well as the COST-TO-GO. However, there is an important drawback in such methods: the set of trajectories that can be obtained is highly dependent on the environment configuration and thus, we may bias the search space into only a subset of it.

Since we need a set of paths that cover as much of the solution space as posible, we propose to overcome this problem by introducing randomness into the steering function, the ESFM. The resultant force

$$\boldsymbol{f}_r = \beta_1\, \boldsymbol{f}_r^{goal} + \beta_2\, \boldsymbol{f}_{people}^{int} + \beta_3\, \boldsymbol{f}_{obstacles}^{int} \tag{3}$$

determines the generated trajectory, being $\boldsymbol{\beta} = [\beta_1, \beta_2, \beta_3]$ a set of random variables sampled as part of the function in Alg. 1, line 5. Sampling $\boldsymbol{\beta}$ enriches the coverage of a higher area in the solution space, that is, we can obtain more distinct solutions by introducing random parameters, which is very usefull for the selection of the best path.

V. MULTI-OBJECTIVE COST-TO-GO FUNCTION

Commonly, cost functions are built after a direct weighted-sum of different variables, that may be expressed in different units, and they are projected into the real space as a scalarization $J : \mathbb{R}^n \rightarrow \mathbb{R}$. Our previous work [1] followed a similar approach: at the end, it becomes a problem of tunning parameters and that solution only works on limited scenarios, which is a very sensitive and non-robust solution.

There are multiple objectives to be minimized in dynamic planning, and we use different and independent criteria instead of a single-objective composed of different variables:

$$\boldsymbol{J}(S, s^{goal}, U) = [J_1, J_2, \ldots, J_I]. \tag{4}$$

As stated before, we aim to obtain a navigation algorithm that considers different cost functions, and combines them independently of the configuration of the scene. In this subsection these costs functions are defined. The cost to reach a goal J_d is

$$J_d(S, s^{goal}) = \sum_{t=t_{ini}}^{t_{end}} \|\boldsymbol{x}_r(t) - \boldsymbol{x}^{goal}\|^2, \tag{5}$$

where we obtain the accumulated square distance value from the initial state at time t_{ini} in the set of states S, to the final state at time t_{end}. The cost orientation J_{or} expresses the difference between the desired orientation and the current orientation

$$J_{or}(S, s^{goal}) = \sum_{t=t_{ini}}^{t_{end}} \|\theta_r(t) - \theta^{goal}\|^2, \tag{6}$$

representing the accumulated distance to the desired goal orientation θ^{goal}. We additionally measure the cost associated to the robot control J_r in the following way

$$J_r(U) = \sum_{t=t_{ini}}^{t_{end}} \|u_r(t)\|^2, \tag{7}$$

that sums the robot inputs u_r throughout the calculated trajectory. Similarly, we define the cost function for the pedestrians J_p as

$$J_p(U) = \sum_{t=t_{ini}}^{t_{end}} \sum_{i=1}^{N} \|u_{p_i}(t)\|^2, \tag{8}$$

where the inputs u_{p_i} are due to the robot influence to other pedestrians while they walk towards their goals. We also take into account the cost produced by nearby obstacles to the robot J_o as

$$J_o(U) = \sum_{t=t_{ini}}^{t_{end}} \sum_{i=1}^{O} \|u_{o_i}(t)\|^2, \tag{9}$$

where again we consider the perturbations to the robot as a result of nearby obstacles u_{o_i}, since we want to avoid collisions.

Our main purpose is to combine these cost functions into a well-posed function. To this end, we are inspired in multi-objective optimization techniques to solve the problem of

finding a set of optimal solutions. The scaling effect of the simple weighted-sum method can be avoided by normalizing the objective functions according to:

$$\bar{J}_i(X) = erf\big(\frac{x-\mu_x}{\sigma_x}\big). \tag{10}$$

We will refer it as the normalized weighted-sum method.

The variables μ_x, σ_x are estimated after the tree $\mathcal{T}$ is built. The multi-objective cost function becomes a single-objective by applying three steps: calculation of each cost $\boldsymbol{J} : \mathcal{S} \times \mathcal{U} \rightarrow \mathbb{R}^I$, a normalization to $(-1, 1)$, and a projection via a weighted sum $J : \mathbb{R}^I \rightarrow \mathbb{R}$:

$$J(S, s_{goal}, U) = \sum_i w_i \bar{J}_i(S, s_{goal}, U). \tag{11}$$

The final cost function is the accumulated cost J throughout the path, where each cost has been correctly normalized and combined according to the $\boldsymbol{w}$ parameters. An interesting consequence of the proposed normalized weight-sum is that the navigation algorithm presents similar performance for very different scenarios, that is, we have carried out all the experiment without changing $\boldsymbol{w}$ to different situations.

A. Cost-to-go

Euclidean distance is a good metric in a geometrical planning problem, nevertheless, it has proven to be less efficient in certain planning schemes while cost-to-go functions work better in kinodynamic planning, as we will show later. We have defined a multi-objective cost function that can also be used to evaluate the cost-to-go from a state to another state.

Most of the works [15], [16], [17] calculate the cost-to-go in the absence of obstacles. In our case, we consider dynamical obstacles and include these interactions, thanks to the ESFM. In Fig. 2 is drawn the process to calculate the nearest vertex in the search tree. We extend a virtual path towards the goal q_{new} from each of the vertices, and calculate the accumulated cost to reach it plus the previous cost.

We will discuss in Sec. VI the coverage of the space explored using a cost-to-go function to find the nearest vertex and we will compare it to a Euclidean approach.

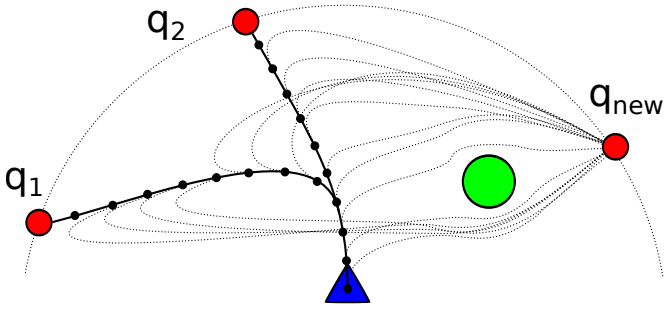

Fig. 2. Scheme of the cost-to-go function to reach q_{new} from each of the vertices in the tree. Each trajectory (dotted lines) is generated using (1),(2), and provides the accumulated cost for each criterion, and thus is named cost-to-go.

VI. Experiments

We have divided the experiments in three parts. We firstly learn the navigation parameters $\boldsymbol{w}$ in a simulated environment. In the second part, we show the importance of a correct cost-to-go function, compared to a Euclidean metric, and in the final part, we evaluate the general performance of the improved algorithm.

The simulation framework used is a custom project built around ROS. Human motion is simulated using the work presented in [3], where we generate people tracks, that is, a sequence of positions over time with constant *id*, and model them as random variables. The robotic platform is simulated according to a unicycle model that is controlled by the AKP.

We have carried out all the simulations in a Intel Core2 Quad CPU Q9650 @ 3.00GHz and memory 3.8 GiB, at an average rate of 5Hz, which is thanks to the steering heuristic. The hardware used for simulations is similar to the PC onboard the real robot. The simulated scenarios are as follows: the robot receives a query to a goal, in different scenarios that are built combining a different number of obstacles and pedestrians walking in the area (see attached video or at `www.iri.upc.edu/groups/lrobots/akp/iros2015.mp4`).

A. Parameter learning

In order to correctly characterize the effect of the $\boldsymbol{w} = [w_d, w_{or}, w_r, w_p, w_o]$ parameters, we have used a Monte Carlo approach to sample the weights $\boldsymbol{w}$, carrying out more than 20k simulations. We have used many different scenarios which consist of a variable number of people and obstacles, and we have calculated the costs associated to the sampled $\boldsymbol{w}$ parameters for each configuration.

The results of the simulations have been averaged in order to address the fact that the scenario is dynamic and the outcome for the same set of parameters can be different depending on the initial conditions, those are, the position of the simulated pedestrians and their corresponding goals.

In Fig. 3 is depicted the expectation for two costs results (distance and obstacles) depending on the same parameter w_{obs}. As it can be seen, there is not a clear value for the weight cost that can minimize simultaneously the obstacles' and people's costs. In addition, there is a high variance in the results, since we are testing a highly dynamic environment.

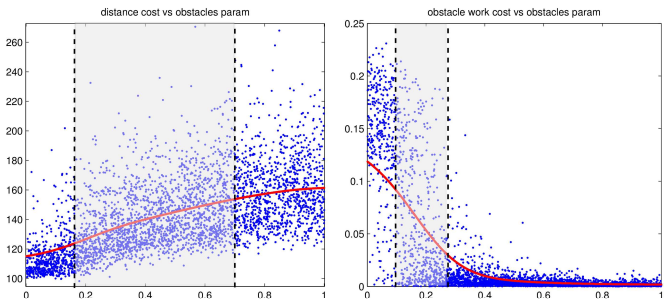

Fig. 3. On the *left* distance cost and on the *right* obstacle cost w.r.t. the w_{obs} weight parameter. Blue dots correspond to single experiments and the red line is the calculated expectation using a Gaussian likelihood. In both graphics appear the interval of the *acceptable* region, intersection around $w_{obs} = 0.2$.

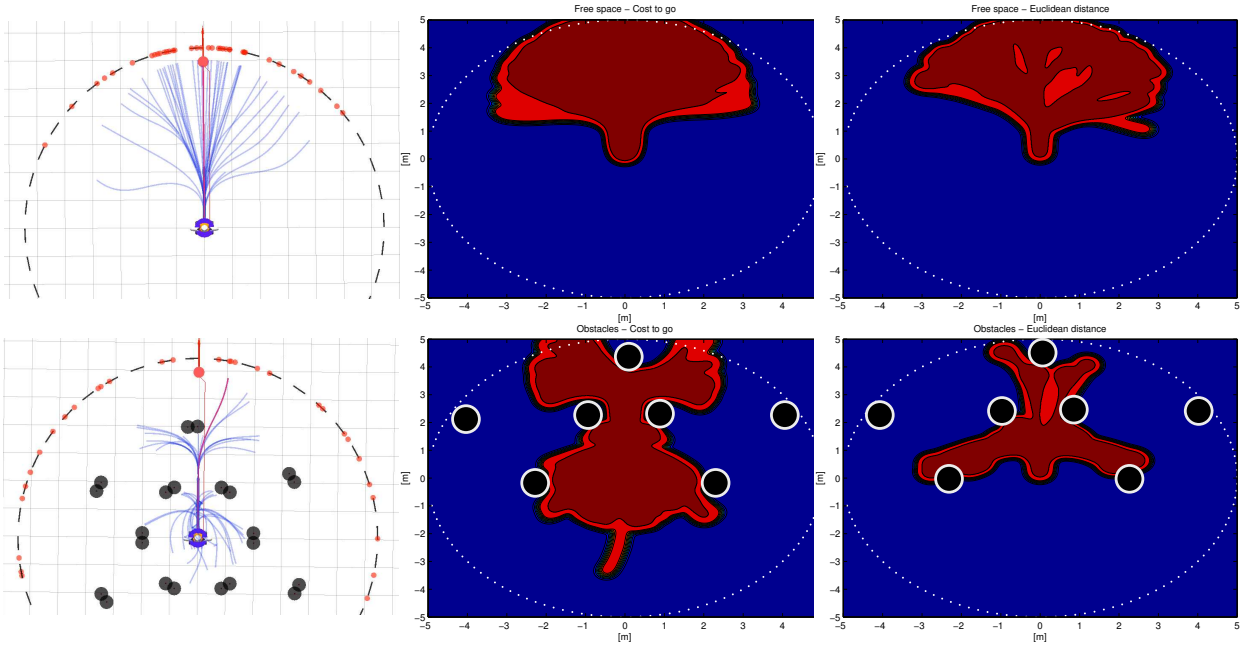

Fig. 4. Coverage of the workspace according to the cost-to-go and randomness in the ESFM, and the Euclidean metric. In red, typically visited areas averaged over a large number of iterations, and in blue, typically non visited regions. The obstacles are plotted as black circles.

For these reasons, we have used an heuristic method to select which are the most convenient values of $\boldsymbol{w}$. Each of the costs considered in this work are drawn as a function of a weight parameter, in the case of Fig. 3, the parameter is w_{obs}. Then, we describe an *acceptable* region starting at 80% of the maximum cost value to the 20% of the minimum value. The intersection, if possible, of the different regions leads to an approximate value for the parameter, after some adjusting by try and error.

Accordingly, we value the parameters and we will use them unchanged for the remaining of the work $w_{distance} = 0.7$, $w_{orientation} = 0.4$, $w_{robot} = 0.5$, $w_{ppl} = 0.3$ and $w_{obs} = 0.2$.

B. Coverage of the solution space

In Fig. 4-*left* are depicted two of the scenarios chosen, to illustrate the problem. On the top, there is an obstacle free scenario and on the bottom an scenario with some obstacles. The tree generated by the Euclidean function, as formulated in [1], appears in Fig. 4-*right* and the cost-to-go metric and randomness, in Fig. 4-*center*.

The area corresponding to visited regions (in red) and the non-visited regions (in blue). It is not possible to generate paths that can visit all the regions, since there is a strong horizon time restriction $h = 5s$. We observe that there are no significant differences in the obstacle free scenario. Both approaches visit a similar area. However, in the scenario with some obstacles there is a great difference: the Euclidean approach presents more difficulties to scape the obstacle area. The cost-to-go and randomness improvements make possible to cover more area and reach the goal.

In order to quantify it, we have run another experiment that measures the area visited in the obstacle scenario, while growing the number of vertices in the AKP. The results are depicted in Fig. 5. Clearly, the cost-to-go and randomness approach outperforms the Euclidean distance, at least under this configuration.

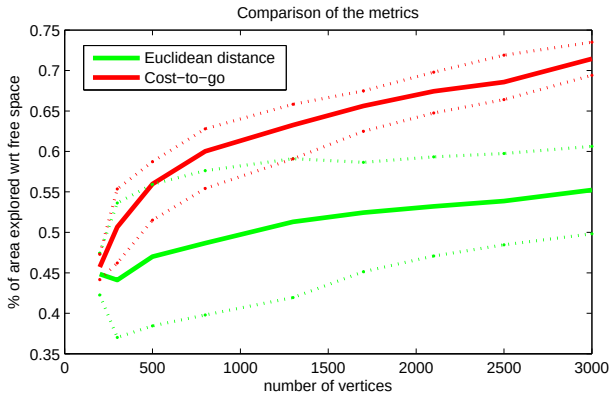

Fig. 5. Coverage of the workspace with respect to the number of vertices used in a scenario with obstacles. Results normalized to the coverage of a free obstacle scenario. Dotted lines are the standard deviations.

In Fig. 6 we can observe the distribution of the cost parameters, prior to normalization, and the corresponding normalization function. The magnitudes of the variables are too different, and any attempt to scale them and sum (former approach) represents a real challenge and an ill-posed problem.

C. Performance

We have performed more than $5k$ experiments in the above described challenging scenarios, comparing our approach, the AKP with cost-to-go, with our former method, the AKP with Euclidean metrics and a third approach: a pure reactive planning [21]. We have set the number of vertices to $K = 800$ for both AKP approaches, and the planner was able to provide a path at a rate not lower than 5Hz.

All the objective cost functions are plotted in Fig. 7. Both of our approaches clearly outperform in all objectives the *reactive* approach, which has demonstrated to behave badly

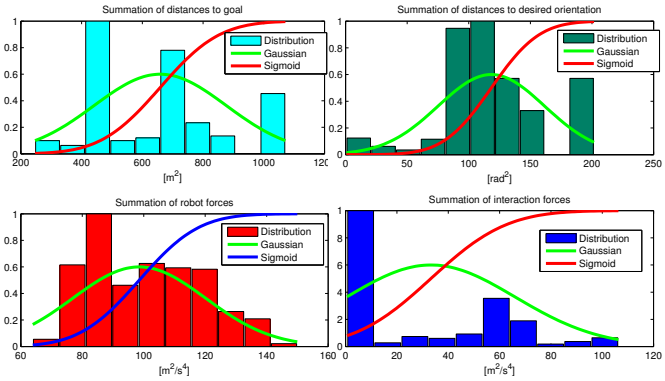

Fig. 6. Distributions of the cost parameters J_i. Adaptively normalizes the different cost values into $(-1, 1)$.

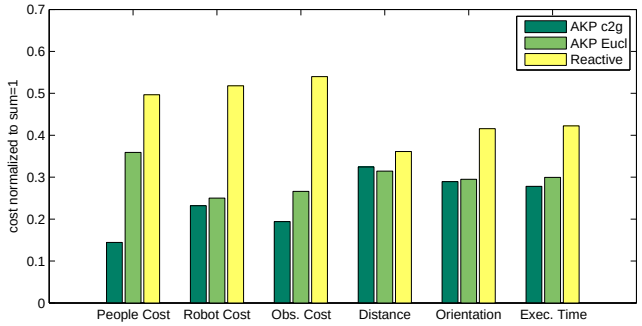

Fig. 7. Results for the different cost functions J_i for the AKP using cost-to-go function, compared with the AKP with Euclidean metrics, and a pure reactive approach. All results have been normalized to sum 1 for visualization reasons.

in dynamic environments, since we often observed a *go-stop-go* strategy. Both AKP behave similarly in simple scenarios, however the AKP with Euclidean metric presents a more "straight behavior" towards its goal, specially in complicated environments (lots of obstacles). Under this circumstances is where the AKP with cost-to-go really shines; it is able to adapt to more challenging environments using the same set of parameters $\boldsymbol{w}$, thanks to the normalized weighted-sum cost, and provide a solution in more challenging environments that the ones used in the learning stage.

Moreover, we have conducted some tests of the AKP in a real robot platform, as shown in the attached video, where we carried out *go-to* queries in a real scenario while people were moving freely.

VII. Conclusions

In addition to our past contributions to robot navigation [1], we have overcome past limitations and complete our approach. One of the main contributions of the present paper is the formulation of a multi-objective cost function when there are several objectives contributing to the robot navigation simultaneously. We have demonstrated that a direct weighted-sum of costs is not able to provide a general solution for different scenarios, while our approach, after a normalization and a correct weighting, is able to provide a set of parameters valid for many scenarios.

Additionally, we have used this cost function to find the nearest vertex in the tree $\mathcal{T}$, based on a cost-to-go metric which we demonstrate to greatly enhance the area covered by the AKP compared to a classical Euclidean distance.

References

[1] G. Ferrer and A. Sanfeliu, "Proactive kinodynamic planning using the extended social force model and human motion prediction in urban environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1730–1735.

[2] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[3] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1688–1694.

[4] M. Kuderer, C. Sprunk, H. Kretzschmar, and W. Burgard, "Online generation of homotopically distinct navigation paths," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014, pp. 6462–6467.

[5] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[6] K. Deb, "Multi-objective optimization," in *Search methodologies*. Springer, 2014, pp. 403–449.

[7] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: the case for cooperation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013, pp. 2145–2152.

[8] G. Ferrer and A. Sanfeliu, "Bayesian human motion intentionality prediction in urban environments," *Pattern Recognition Letters*, vol. 44, pp. 134–140, 2014.

[9] ——, "Behavior estimation for a complete framework of human motion prediction in crowded environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 5940–5945.

[10] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Probabilistic motion planning among moving obstacles following typical motion patterns," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 4027–4033.

[11] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 874–883, 2007.

[12] M. Svenstrup, T. Bak, and H. J. Andersen, "Trajectory planning for robots in dynamic human environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4293–4298.

[13] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[14] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[15] E. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 5021–5028.

[16] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2012, pp. 2537–2542.

[17] B. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3713–3719.

[18] C. Park, J. Pan, and D. Manocha, "Real-time optimization-based planning in dynamic environments using gpus," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 4090–4097.

[19] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1928–1935.

[20] L. Palmieri and K. O. Arras, "A novel rrt extend function fo efficient and smooth mobile robot motion planning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 205–211.

[21] G. Ferrer, A. Garrell, and A. Sanfeliu, "Social-aware robot navigation in urban environments," in *European Conference on Mobile Robotics, ECMR.*, 2013, pp. 331–336.