Learning the Hidden Human Knowledge of UAV Pilots when navigating in a cluttered environment for improving Path Planning

Ignacio Alzugaray and Alberto Sanfeliu[1]

***Abstract*— We propose in this work a new model of how the hidden human knowledge (HHK) of UAV pilots can be incorporated in the UAVs path planning generation. We intuitively know that human's pilots barely manage or even attempt to drive the UAV through a path that is optimal attending to some criteria as an optimal planner would suggest. Although human pilots might get close but not reach the optimal path proposed by some planner that optimizes over time or distance, the final effect of this differentiation could be not only surprisingly better, but also desirable. In the best scenario for optimality, the path that human pilots generate would deviate from the optimal path as much as the hidden knowledge that its perceives is injected into the path. The aim of our work is to use real human pilot paths to learn the hidden knowledge using repulsion fields and to incorporate this knowledge afterwards in the environment obstacles as cause of the deviation from optimality. We present a strategy of learning this knowledge based on attractor and repulsors, the learning method and a modified RRT* that can use this knowledge for path planning. Finally we do real-life tests and we compare the resulting paths with and without this knowledge.**

I. INTRODUCTION

Rescue missions in natural disasters or accidents in industrial plants usually involve inspection tasks to evaluate the level of damage of the injured items. While these tasks might be hazardous for humans or their nature make them unfeasible in relatively quick time, an UAV might be sent to evaluate the losses. Although its flight may be supervised by a security pilot, it might be required to fly autonomously, in which case path planning algorithms become necessary.

Most of the path planners used nowadays for guiding UAVs [1] are based on RRT [2] due to their great performance in terms of computational time, low parametrization and efficiency [3][4]. Such planners are usually based on a minimum-distance criteria so that the solution converges to an optimal or approximately optimal minimum distance path. Although the generated paths might effectively avoid obstacles in the environment, attempting to use such planners for the aforementioned tasks could be risky: if the generated path is such that the UAV passes very close, for example, to high-temperature gas line leaks, fire or any other source of danger, the integrity of the robot may result damaged.

Although the humans might be aware of these risks, the aforementioned path planners do not typically include this knowledge during path generation. Instead, they generate a path by minimizing some other features, i.e. number of nodes, number of iterations or distance to the goal.

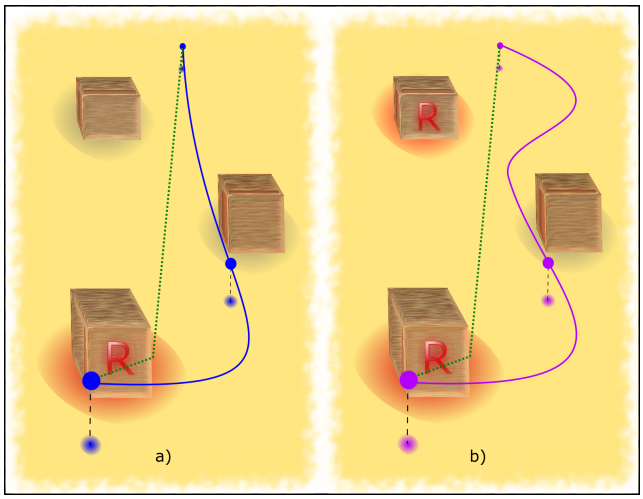


Fig. 1. In a) the blue line represents the human pilot path when guiding an UAV. The green line represents the generated RRT path without taking into account the HHK. One of the environment object (the one with the big red 'R') entails risk. The pilot is aware of it and thus deviates largely the path in order to separate the UAV from the risk. In b) the magenta line represents the generated RRT path including the HHK learned by means of the blue path in a). Additionally, the learned risk level is extrapolated to another object in the scene, which the human knows that has the same level of hazardous than the first one. In consequence, the UAV automatically avoids passing close to that object as well.

The knowledge that a human can provide about the environment could be not only valuable, but sometimes strictly necessary for ensuring the success of the mission and for keeping the robot intact. This knowledge could be learned if a human pilot, who is aware of the risks, guides the UAV to a local destination avoiding with sufficient security the risky obstacles or areas. Specifically, the risk level of each dangerous item could be learned individually and then use this prior knowledge to label further items in the scene that are known to have a similar level of riskiness. See Figure 1.

In this paper we propose a method that learns the HHK when they perform a path by teleoperating the robot. The learned knowledge can the adapted and applied to a classical planner in order to modify the solution such that the resulting path looks similar to a human path. The proposed method is in between the path planning and the learning by demonstration method framework. This method embeds the learned knowledge from the humans in the map in which the robot navigates. Using this approach, the knowledge can

*This work was supported by the Spanish Ministry of Science and Innovation, project Rob-Int-Coop DPI2013-42458-P and AEROARMS European project H2020-ICT-2014-1-644271

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

be reused with independence of the chosen planner prior adaptation.

The idea behind the method is that the HHK when performing an UAV path can be transferred to the environment obstacles, in such a way that the obstacles are repulsion fields, represented as potential field functions, that deviate the planned path in a human-like path (see Figure 1). What we learn from the human path is the modification of the repulsion parameters to adjust the planned path to the human one. These parameters are learned using real trajectories generated by a human pilot guiding an UAV through a cluttered environment. Although we know that this is a risky assumption, we will prove through the real life experiments that could be a good approximation.

In this work we are applying these ideas using a RRT planner, which takes into account the influence of the objects in the planned path. There exist several path planners able to consider an explicitly given analytic function as the one that we propose. For instance, in [5] the exploratory strength of the RRT algorithm is combined with the flexibility of a cost function defined over the configuration space such that the robot is able to navigate through its valleys and saddle points. We can easily think on our potential field function as the cost function required by [5] and use this planner in order to simulate the trajectories needed for our learning process. Other methods that combine RRT algorithms with potential fields are proposed in [6][7]. In the latter, the potential field is used as a cost function in order to penalize those nodes generated by an RRT planner that are far away from the valleys of the function.

In the following sections we first explain the general approach, then the obstacle representation and the repulsion field. In section III we describe the learning algorithm, describing the identification of the relevant repulsors, the interaction between them and the optimization procedure to obtain the best parameters. In section IV we explain the adaptation of a RRT based algorithm to take into account the influence of the repulsion fields. In section V we describe the experiments using two different UAV pilots that teleoperates the UAVs from different starting points, but following a similar path. Finally, in section VI, we summarize the work.

II. HHK LEARNING METHOD

A. General approach

The proposed method learns the HHK when the human pilots are executing a path, using a general strategy based on attractors and repulsors, in a similar way that is used in Learning from Demonstration [8] since a path generated by a human pilot is used as reference to learn the parameters of a model. However, in our case we learn the HHK through the repulsion field related to the environment objects.

To introduce the method, we define that any path $\mathcal{P}$ obtained from a path planner is guided by two elements: the attractors θ_{att} and the repulsors θ_{rep}.

$$\mathcal{P} = f(\theta_{att}, \theta_{rep}) \tag{1}$$

where the f depends strictly on the planner.

The attractor θ_{att} represents the planner strategy to generate a path to the goal location. The repulsors θ_{rep} usually represents the obstacles and prevents the robot to navigate in the zones where collision may happen.

The most straightforward example of this formulation is the potential path planner where θ_{att} and θ_{rep} apply an attractive and repulsive force respectively to the robot at each point. In this case, the repulsors θ_{rep} are explicit in the obstacles and the attractor θ_{att} is the force exerted by the goal (see also [9] in the framework of dynamical movement primitives, DMPs, or [10] in the framework of planning in a human-like manner). In the classic RRT, the repulsors θ_{rep} are implicit in the obstacles to provide obstacle avoidance paths whereas the implicit attractor θ_{att} is the force to reach a solution path toward the goal if it exists. In this paper we present a modified version of the RRT* [3] where the planner is partially guided by the repulsors.

The characteristics of the repulsors are modified accordingly by means of the learning algorithm to capture the HHK. Each of the map repulsors represents as a source of the repulsion field and the global repulsion field is computed combining the effect of the different repulsors. The navigation plan at each point of the path is guided by the repulsion field and the robot navigates following the regions with lowest magnitude in the repulsion field, this is, the valleys.

B. Obstacle representation

In robot navigation, the map is represented by several objects which, in this work, are static obstacles. Each obstacle is defined by one or more repulsors placed in relevant parts of the obstacle such as the centroid or parts of the perimeter.

The characteristics of each repulsor reflects directly the danger of navigating close to this location. However it can also encode other knowledge usually considered by humans such as risk and uncertainty.

Let's denote R_j the j-th repulsor of the set $\mathcal{M} = \{R_1, R_2, \ldots, R_n\}$ which represents the obstacles of the map. The influence of the repulsor R_j is completely defined by the tuple of parameters $\Theta_{R_j} = \left\{A_{R_j}, B_{R_j}, r_{0R_j}, \mathbf{x}_{R_j}\right\}$. The parameters A_{R_j} and B_{R_j} represent respectively the intensity and the decay factor of the repulsion. The repulsion includes a safety area modeled by a ball of radius r_{0R_j} in which the robot cannot navigate. This parameter is set according to the robot size to prevent collision with the obstacles in the map during the navigation. Finally, the parameter $\mathbf{x}_{R_j}$ is the position vector which represents the location of the repulsor in the space.

The repulsion $\mathbf{F}_{R_j}(\mathbf{x})$, in the location $\mathbf{x}$, generated due to the repulsor R_j, can be modeled as a force vector with the following expression:

$$\mathbf{F}_{R_j}(\mathbf{x}) = A_{R_j} \exp\left(-\frac{d_{R_j}(\mathbf{x}) - r_{0R_j}}{B_{R_j}}\right) \frac{\mathbf{x} - \mathbf{x}_{R_j}}{d_{R_j}(\mathbf{x})} \tag{2}$$

where $d_{R_j}(\mathbf{x})$ is the euclidean distance between the point $\mathbf{x}$ and the location of the repulsor $\mathbf{x}_{R_j}$ and the last term specifies the orientation of the force vector.

C. Repulsion field

The set $\mathcal{M}$ is composed by several repulsors which model the influence of the obstacles in the map. The repulsion field is obtained by computing the influence of these repulsors. The learning of the set of HHK repulsor parameters depends on how this repulsion field is defined.

Let's denote $\mathcal{M}_{\mathbf{x}} \subseteq \mathcal{M}$ the subset of visible repulsors from the point $\mathbf{x}$, i.e. the ones that are in the direct line of sight from $\mathbf{x}$. Then the repulsion field $\mathcal{F}$, evaluated in the point $\mathbf{x}$, is defined as a scalar function using $F_{R_j}(\mathbf{x}) = \|\mathbf{F}_{R_j}(\mathbf{x})\|$ in the following expression:

$$\mathcal{F}(\mathbf{x}) = \max_{R_j} F_{R_j}(\mathbf{x}), \quad \forall R_j \in \mathcal{M}_{\mathbf{x}} \tag{3}$$

The projections of the set of points of the valleys in the repulsion field, denoted by $\mathcal{V}_{R_i R_j}$, are the ones that satisfy the following:

$$\mathcal{V}_{R_i R_j} = \left\{ \mathbf{x} \;\middle|\; \begin{array}{c} F_{R_i}(\mathbf{x}) = F_{R_j}(\mathbf{x}) \\ \wedge \\ F_{R_i}(\mathbf{x}) \geq F_{R_k}(\mathbf{x}), \quad \forall R_k \in \mathcal{M}_{\mathbf{x}} \end{array} \right\} \tag{4}$$

All the projections of the valleys which were generated due to the influence of the repulsor R_j, define the repulsor border ∂_{R_j}.

$$\partial_{R_j} = \bigcup \mathcal{V}_{R_j R_k}, \quad \forall R_k \in \mathcal{M}, \quad j \neq k \tag{5}$$

Finally, we define Ω_{R_j} as the region in which the repulsor R_j applies the greatest repulsion force compared to all the visible repulsors.

$$\Omega_{R_j} = \left\{ \mathbf{x} \,\middle|\, F_{R_j}(\mathbf{x}) > F_{R_k}(\mathbf{x}), \quad \forall R_k \in \mathcal{M}_{\mathbf{x}} \right\} \tag{6}$$

Note that ∂_{R_j} does not represent the boundary of the region Ω_{R_j} due to the discontinuities in the mathematical definition of the repulsion field.

III. LEARNING ALGORITHM

The proposed method assumes that the HHK is distributed in two parts when navigating with a robot, one included in the path planner and the other one in the obstacles in the map. In this paper we will focus on learning the repulsors parameters, $\Theta_{R_j} = \{A_{R_j}, B_{R_j}, r_{0R_j}, \mathbf{x}_{R_j}\}$, that model these obstacles using human pilot paths as reference. Therefore the information embedded in the obstacles can be reused when planning different new paths in the same map.

The proposed algorithm learns the parameters Θ_{R_j} by fitting iteratively the valleys of the defined repulsion field into the pilot paths. In each learning iteration the algorithm first identifies which repulsors have to be modified. Afterwards this modification is obtained by means of an optimization problem posed from an overconstrained system of equations that represents the constraints between pairs of repulsors.

In order to simplify the learning process, the algorithm is limited to learn only the parameter B_{R_j} (instead of B_{R_j} and A_{R_j} at the same time) while maintaining the other parameters equal and constant for all the repulsors. Usually r_{0R_j} is a constant value related to the dimensions of the robot and $\mathbf{x}_{\mathbf{R_j}}$ depends on the environment.

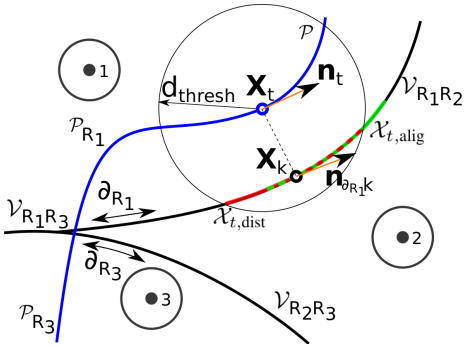


Fig. 2. Identification of applicable repulsors.

A. Identification of applicable repulsors

The influence of the repulsors are limited to a local area. This implies that the different segments of the pilot path can be used to learn the parameters related to a subset of repulsors. The following procedure identifies which are the applicable repulsors in each of the learning iterations.

Let's denote $\mathcal{P} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ the human pilot path and let us define the subset $\mathcal{P}_{R_j} = \mathcal{P} \cap \Omega_{R_j}$. Then for each point $\mathbf{x}_t \in \mathcal{P}_{R_j}$ we establish a correspondence to a point in the border $\mathbf{x}_k \in \partial_{R_j}$ using the following two conditions.

a) Distance condition: Only the points in the nearby of the point $\mathbf{x}_t$ are considered, i.e. the points that are within a ball of radius d_{thresh} centered in the point $\mathbf{x}_t$.

$$\mathcal{X}_{t,\text{dist}} = \left\{ \mathbf{x}_k \in \partial_{R_j} \;\middle|\; \|\mathbf{x}_k - \mathbf{x}_t\| \leq d_{\text{thresh}} \right\} \tag{7}$$

b) Alignment condition: Let $\mathbf{n}_t$ denotes the unitary vector tangent to the segment $\mathcal{P}_{R_j}$ at the point $\mathbf{x}_t$. Equivalently, $\mathbf{n}_{\partial_{R_j} k}$ is defined as the tangent to the boundary ∂_{R_j} at the point $\mathbf{x}_k$. Then, only the points $\mathbf{x}_k$ such that human path and boundary direction are aligned compared with a threshold α_{thresh} are considered.

$$\mathcal{X}_{t,\text{alig}} = \left\{ \mathbf{x}_k \in \partial_{R_j} \;\middle|\; \left| \mathbf{n}_i \cdot \mathbf{n}_{\partial_{R_j} k} \right| \leq \alpha_{\text{thresh}} \right\} \tag{8}$$

From all the points in the border that satisfy both conditions, this is, $\mathcal{X}_t = \mathcal{X}_{t,\text{dist}} \cap \mathcal{X}_{t,\text{alig}}$, the closest point to the path point $\mathbf{x}_t$ is selected (see figure 2).

$$\mathbf{x}_k = \arg\min_{\mathbf{x}} \|\mathbf{x} - \mathbf{x}_t\|, \quad \mathbf{x} \in \mathcal{X}_t \tag{9}$$

If $\mathcal{X}_t \neq \emptyset$, there is a correspondence between the point $\mathbf{x}_t \in \mathcal{P}_{R_j}$ and the point $\mathbf{x}_k \in \partial_{R_j}$. Since $\mathbf{x}_k \in \mathcal{V}_{R_i R_j} \subseteq \partial_{R_j}$, the point $\mathbf{x}_t$ has to be considered for the correction of the learning of the repulsors R_i and R_j. In case $\mathcal{X}_t = \emptyset$ no correspondence has been found and the point $\mathbf{x}_t$ is not considered in the following steps of the learning algorithm.

B. Interaction between a pair of repulsors

Defining $\mathbf{x}_{R_i R_j} \in \mathcal{T}_{R_i R_j} \subseteq \mathcal{P}$ as a point such that a correspondence to another point $\mathbf{x}_k \in \mathcal{V}_{R_i R_j}$ has been found using the previous procedure. Then the set of n points in $\mathcal{T}_{R_i R_j}$ are respectively used to learn the parameters B_{R_i} and B_{R_j} of the repulsors R_i and R_j.

Let $\hat{d}_{R_j,\mathcal{T}_{R_iR_j}}$ be the mean corrected distance from the repulsor R_j defined as

$$\hat{d}_{R_j,\mathcal{T}_{R_iR_j}} = \frac{1}{n} \sum_{\mathbf{x}_{R_iR_j} \in \mathcal{T}_{R_iR_j}}^{n} d_{R_j}(\mathbf{x}_{R_iR_j}) - d_{0R_j} \quad (10)$$

The location of the valley projection $\mathcal{V}_{R_iR_j}$ can be locally modified by only using the first constraint of the definition (4), i.e. $F_j(\mathbf{x}) = F_i(\mathbf{x})$. The proposed method modifies only the decay ratio parameter B for that purpose and the other parameters are maintained constant and equal for all the repulsors. Using the model of the repulsors expressed in (2) and the mean corrected distance, the following expression can be obtained:

$$\hat{d}_{R_j,\mathcal{T}_{R_iR_j}} B_{R_i} = \hat{d}_{R_i,\mathcal{T}_{R_iR_j}} B_{R_j} \quad (11)$$

Using this expression the relation between the B parameters of the pair of repulsors can be computed so that if it is satisfied, the resulting valley projection $\mathcal{V}_{R_iR_j}$ lies in the nearby of the points $\mathbf{x}_{R_iR_j} \in \mathcal{T}_{R_iR_j}$.

C. Optimization

When the method is applied to the complete pilot path, we have several expressions (11) that relate different pairs of repulsors. The simultaneous fulfillment of all these multiple interconnected relations is usually impossible as they represent an overconstrained system of equations. To solve it, the solution of the system can be posed as an optimization problem where the aim is to compute the parameter B of the repulsors.

For each relation between a pair of repulsors R_i and R_j an expression is defined such as

$$f_{R_iR_j}(B_{R_i}, B_{R_j}) = w_{R_iR_j}(\hat{d}_{R_j,\mathcal{T}_{R_iR_j}} B_{R_i} - \hat{d}_{R_i,\mathcal{T}_{R_iR_j}} B_{R_j})^2 \quad (12)$$

where $w_{R_iR_j}$ is a weight related to the importance of the relation between the repulsors R_i and R_j, computed as the number of considered n points, when obtaining $\hat{d}_{R_j,\mathcal{T}_{R_iR_j}}$ in (10).

All the different relations can be stacked in the same function

$$f(\mathbf{\Theta}) = \sum f_{R_iR_j}(B_{R_i}, B_{R_j}), \quad \forall R_i, R_j \in \mathcal{M} \quad (13)$$

with the vector of parameters $\mathbf{\Theta} = [B_{R_1}, B_{R_2}, \ldots, B_{R_m}]^T$.

Additionally, another function is optimized in order to maintain the values of the B parameters as low as possible.

$$g(\mathbf{\Theta}) = \sum_{i}^{m} B_{R_i}^2 \quad (14)$$

The optimization function is defined as

$$J(\mathbf{\Theta}) = \alpha f(\mathbf{\Theta}) + \beta g(\mathbf{\Theta}) \quad (15)$$

where $\alpha \gg \beta$ to prioritize the fulfillment of the defined relations.

Since the expression J is composed by squared terms and squared differences, it can be rearranged so that:

$$J(\theta) = \mathbf{\Theta}^T \mathbf{Q} \mathbf{\Theta} \quad (16)$$

where $\mathbf{Q}$ matrix is composed by diagonal and off-diagonal elements corresponding respectively to the terms multiplying the squared parameters and the ones multiplying the cross product between different parameters.

Using this expression, the optimization process can be posed as a quadratic programming problem. Several optimizers can be applied to solve this problem in which we include the bound $B \in [B_{\text{min}}, B_{\text{max}}]$. The initial value of the B parameter in the repulsors is also set according to B_{min}.

The learning algorithm is proposed as an iterative procedure in which the repulsor field is to be gradually modified. Therefore, instead of modify the values of the B parameters according to the results of the proposed optimization problem, the new values are obtained restricting the variation of the parameter value between successive iterations.

Let us define $B_{R_j}^{k)}$ as the value of the decay ratio of the repulsor R_j and as $B_{R_j}^{*k)}$ the value obtained by means of the optimization in the iteration k. The parameter for the next iteration is computed as follows:

$$B_{R_j}^{k+1)} = B_{R_j}^{k)} + \max\left(B_{R_j}^{*k)} - B_{R_j}^{k)}, \overline{\Delta B}\right) \quad (17)$$

where $\overline{\Delta B}$ is a restriction in the change of the parameter B between successive iterations. The $\overline{\Delta B}$ parameter is directly related to the learning rate of the algorithm. Nonetheless, it is convenient to set this parameter to a low value so that the number of points of the pilot path used in each iteration of the algorithm (see section III-A) varies gradually.

IV. RRT* ADAPTATION

The aforementioned learning algorithm is able to learn the parameters of the respulsors to capture the HHK. Then this information can be used in any path planner as long as the influence of the obstacles, i.e. the repulsors, can be taken into account. In this paper, we have adapted the RRT* [3] for this purpose.

Based on the original RRT algorithm, the RRT* defines a cost function that is used to evaluate the different nodes. The cost of a new node is computed when it is connected to the parent node in the current tree, which includes the cost of the parent node and the cost of the connection. The possible changes in the current connections of the tree are later evaluated in the rewiring step towards probabilistic optimality. In our approach, we include in the cost function the influence of the repulsors.

Let $\mathbf{x}_c$ be the new sample that will be connected to the current tree using the parent node $\mathbf{x}_p$. The cost of the parent node is specified by $c(\mathbf{x}_p)$. Let's denote $c^*(\mathbf{x}_c, \mathbf{x}_p)$ the connection cost between the nodes $\mathbf{x}_c$ and the tree node $\mathbf{x}_p$. Then the cost of the new node $\mathbf{x}_c$ is computed as $c(\mathbf{x}_c) = c(\mathbf{x}_p) + c^*(\mathbf{x}_c, \mathbf{x}_p)$. The cost of the connection c^* is defined as follows:

$$c^*(\mathbf{x}_c, \mathbf{x}_p) = (1-\gamma) c_d(\mathbf{x}_c, \mathbf{x}_p) + \gamma c_r(\mathbf{x}_c) \quad (18)$$

where $c_d(\mathbf{x}_c, \mathbf{x}_p)$ is computed as $c_d(\mathbf{x}_c, \mathbf{x}_p) = \|\mathbf{x}_c - \mathbf{x}_p\|/d_{max}$, being d_{max} the maximum distance in the connection used during the steering and rewiring steps of the

algorithm. The cost due to the repulsors c_r is defined as the evaluation of repulsion field in that point with a normalization factor $c_r(\mathbf{x}_c) = \mathcal{F}(\mathbf{x}_c)/(\max \mathcal{F}(\mathbf{x}))$. Lastly, the parameter $\gamma \in [0, 1]$ weights the relative importance between the length of the path and the influence of the repulsors.

V. EXPERIMENTS

A. Environment and pilot trajectories

In order to show the proposed framework for learning the HHK of the pilots, data from real experiments in a cluttered scenario produced by the team of FADA-CATEC in the EU ARCAS project [11] has been used. Two different pilots teleoperated the UAV following the same path but from different starting points. These paths were captured using an indoor positioning system (VICON).

The cluttered environment can be seen in Figure 3, where the set of repulsors representing the obstacles of the mockup are shown. Each repulsor shows its identifier, safety area corresponding to r_{0R_i} (light blue) and an area of influence proportional to B_{R_i} (dashed green). Linear repulsors representing aligned group of obstacles have each side represented by a different repulsor so that their properties can be described independently. Each repulsor R_j is initialized with default values: $A_{R_j} = 1$ [N], $B_{R_j} = 0.3$ [m] and $r_{0R_j} = 0.25$ [m]. The navigation in the map is constrained through fictitious barriers also represented in the figure (dashed black lines).

The whole set of paths obtained during the experiments were classified in two different scenarios. In the first scenario, represented by the experiment `E1_N`, the pilots simply navigate through the map avoiding the collision with the obstacles. In the second scenario, the pilots were told to consider additional risk regarding the obstacles located in the top or bottom (repulsors R_4 and R_2 in our representation) part of the upper narrow passage represented by the experiments `E3_R` and `E2_R` respectively. The aforementioned paths can be visualized in Figure 3.

B. Learning of the parameters

The proposed learning algorithm is applied to each one of the pilot paths up to 25 iterations. During the identification of applicable repulsors, the thresholds $d_{thresh} = 1.5$ [m] and $\alpha_{thresh} = 0.9$ were used in (7)-(8). The weights in of the optimization function (15) were set to $\alpha = 10$ and $\beta = 1$ with the constraints $B_{\text{min}} = 0.3$ [m] and $B_{\text{max}} = 1.75$ [m]. Additionally, the maximum variation of the decay ratio between iterations in (17) was limited to $\overline{\Delta B} = 0.1$ [m].

Since each experiment contains several paths, the mean of the learned parameters was used. In experiment `E1_N` since there were six different paths, only three of them were used in the learning of the parameters while the others were left for validation. On the other hand, due to the small number of paths in experiments `E2_R` and `E3_R`, all paths were used for learning of the parameters and also for validation of the results.

The mean of the parameters $\overline{B}_{R_i}$ and the maximum difference ΔB_{R_i} to the individual values obtained from each path is collected in Table I. We can observe how the learning algorithm captures the risk in each experiment. In `E1_N` the learned parameters have similar values and close to the minimum B_{min}, since they capture the HHK of paths in which the obstacles are equally avoided. In `E2_R` and `E3_R` the algorithm successfully captures the HHK regarding the risk related respectively to the repulsors R_2 and R_4 (highlighted in Table I). Note that when the HHK is learned, not only is the risk captured in the repulsors but also other navigation features of the expert pilot are obtained as well. These other features include, for instance, the decision of early positioning before entering in the lower narrow passage captured in the repulsor R_3 of `E3_R`.

Experiment	E1_N		E2_R		E3_R	
R_i	$\overline{B}_{R_i}$	ΔB_{R_i}	$\overline{B}_{R_i}$	ΔB_{R_i}	$\overline{B}_{R_i}$	ΔB_{R_i}
R_1	0.30	0.00	0.30	0.00	0.55	0.06
R_2	0.39	0.09	0.87	0.02	0.30	0.00
R_3	0.35	0.08	0.41	0.01	0.59	0.01
R_4	0.47	0.08	0.31	0.01	0.79	0.05
R_5	0.54	0.16	0.36	0.01	0.36	0.00
R_6	0.30	0.00	0.40	0.01	0.39	0.01
R_7	0.43	0.17	0.37	0.03	0.39	0.05
R_8	0.30	0.00	0.30	0.00	0.30	0.00
R_9	0.32	0.05	0.37	0.02	0.31	0.00
R_{10}	0.34	0.05	0.30	0.00	0.30	0.00
R_{11}	0.30	0.01	0.30	0.00	0.31	0.01

TABLE I
LEARNED PARAMETERS.

C. Validation of the results

To validate the results, the repulsors with the learned parameters were applied to the adapted RRT* (we adapted the implementation of [12]). In this case, the weight used in the cost function (18) was set to $\gamma = 0.85$.

We evaluated the resulting path through the area error computed as the mean of the area enclosed between the generated path and all the paths of the pilots in the experiment. This area error is also subdivided into different subareas (1, 2 and 3), where subarea 1 includes the lower narrow passage; 2 includes the transition between the bottom and upper part; and 3 includes the upper narrow passage. In Table II we compare the area error of our modified RRT* considering the learning HHK in the repulsors, with the RRT* considering only the euclidean distance in the cost function.

Experiment	E1_N		E2_R		E3_R	
Area [m^2]	Ours	Eucl.	Ours	Eucl.	Ours	Eucl.
Area 1	1.27	1.85	1.05	2.24	0.75	1.4
Area 2	2.43	5.56	3.64	9.42	1.64	5.08
Area 3	1.90	4.83	1.32	9.54	6.34	10.35
Area $_{\text{total}}$	5.60	12.23	6.01	21.19	8.72	16.82

TABLE II
EVALUATION OF THE RESULTING TRAJECTORIES.

In Figure 3 we can visualize the resemblance of the resulting path in our approach to the pilot paths. Note that

in all the experiments our approach outperforms the RRT* which only considers the euclidean distance.

In E1_N and E2_R, the error is mainly located in Area 2. In this area, the pilot has a vast space without obstacles and therefore he can navigate freely. Since our approach relies on capturing the HHK in the map, i.e. the repulsors, this part of the navigation cannot be completely characterized. A similar situation occurs in Area 3 in E3_R as the pilot chooses to keep a large security distance from the risky obstacles. Our approach performs better in cluttered scenarios where the HHK is captured into several obstacles in the map.

VI. CONCLUSIONS

In this paper we have introduced a novel approach to capture the HHK of human expert pilots using paths of UAVs from real life experiments. This HHK has been successfully captured in the map representation of the environment using repulsors. In the experiments we have also shown how to adapt an RRT* algorithm to include this new information in the path planning and obtain human-like trajectories. The new model has some constraints. It can only be applied to geometric trajectories, not to dynamic ones where velocity and acceleration are taken into account. Moreover the HHK can only be transferred to physical repulsors, but not to whirlwind or other repulsors that may exist. Future work will include these issues and the possibility of discovering non physical repulsors using this model.

REFERENCES

[1] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous uav guidance," *Journal of Intelligent Robotic Systems*, vol. 57, no. 1-4, p. 65100, 2010.

[2] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, technical report TR 98-11, Tech. Rep., 1998.

[3] E. F. S. Karaman, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research, Vol.30, 7, pp.846-894*, 2011.

[4] A. Boeuf, J. Cortes, R. Alami, and T. Simeon, "Planning agile motions for quadrotors in constrained environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014*.

[5] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *Robotics, IEEE Transactions on*, vol. 26, no. 4, pp. 635–646, 2010.

[6] O. Khatib, "Real-time obstacle avoidance for manipulators and fast mobile robots," *International Journal of Robotics Research, vol.5, pp.90*, 1986.

[7] H. A. M. Svenstrup, T. Bak, "Trajectory planning for robots in dynamic human environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010), pp.4293-4298, 2010*.

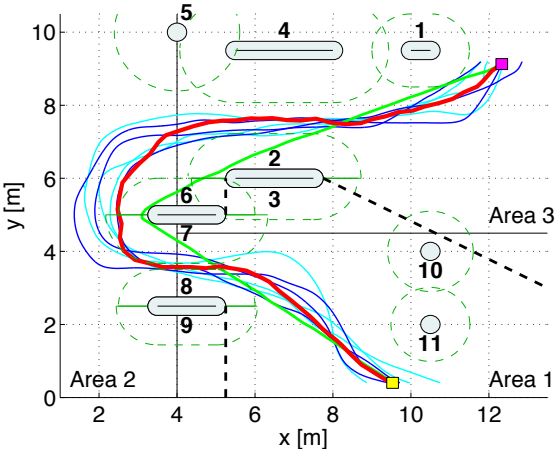
[8] S. Schaal, "Learning from demonstration," *Advances in neural information processing systems*, pp. 1040–1046, 1997.

[9] ——, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive Motion of Animals and Machines*. Springer, 2006, pp. 261–280.

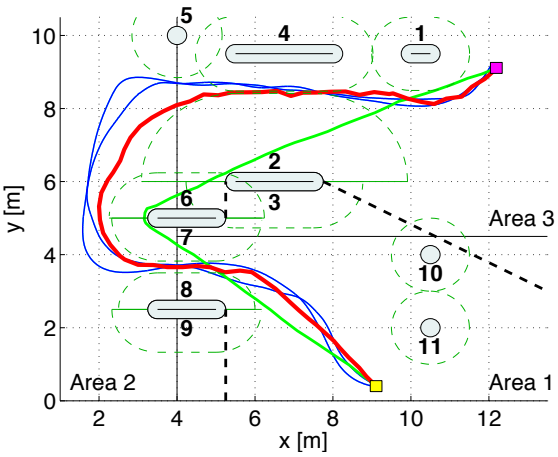
[10] G. Ferrer and A. Sanfeliu, "Proactive kinodynamic planning using the extended social force model and human motion prediction in urban environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp.1730-1735, 2014*.

[11] ARCAS, "Aerial robotics cooperative assembly system," url http://www.arcas-project.eu, 2013-2016.

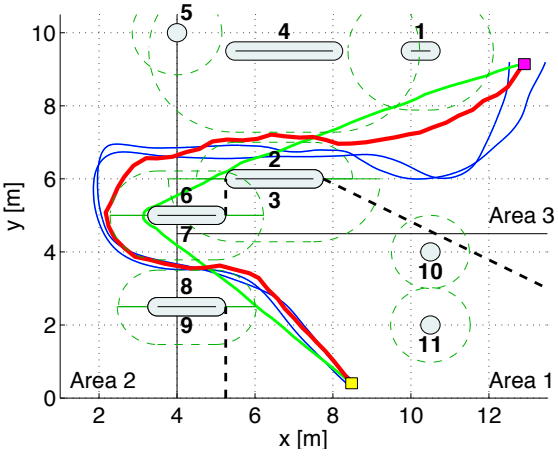
[12] O. Adiyatov and H. Varol, "Rapidly-exploring random tree based memory efficient motion planning," in *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on*, 2013, pp. 354–359.



(a) E1_N



(b) E2_R



(c) E3_R

Fig. 3. Navigating in an environment: without risk (top), with risk on the top and bottom of the upper narrow passage (middle and bottom respectively). Pilot paths used for learning of the parameters (cyan), paths used for validation (blue), RRT* using learned parameters of the repulsors (red), RRT* using euclidean distance (green). The initial and final position are represented by boxes (yellow and magenta respectively).