Task Reallocation in Multi-Robot Formations

Noa Agmon, Gal A Kaminka, Sarit Kraus and Meytal Traub

***Abstract*—This paper considers the task reallocation problem, where k robots are to be extracted from a coordinated group of N robots in order to perform a new task. The interaction between the team members and the cost associated with this interaction are represented by a directed weighted graph. Consider a group of N robots organized in a formation. The graph is the monitoring graph which represents the sensorial capabilities of the robots, i.e., which robot can sense the other and at what cost. The team member reallocation problem with which we deal, is the extraction of k robots from the group in order to acquire a new target while minimizing the cost of the interaction of the remaining group, i.e., the cost of sensing amongst the remaining robots. In general, the method proposed in our work shifts the utility from the team member itself to the interaction between the members, and calculates the reallocation according to this interaction cost. We found that this can be done optimally by a deterministic algorithm, while reducing the time complexity from $\mathcal{O}(N^k)$ to $\mathcal{O}(2^k)$, thus resulting in a polynomial time complexity in the common case where a small number of robots is extracted, i.e., when $k = \mathcal{O}(N)$. We show that our basic algorithm creates a framework that can be extended for use in more complicated cases, where more than one component should be taken into consideration when calculating the robots' cost of interaction. We describe two such extensions: one that handles prioritized components and one that handles weighted components. We describe several other non-robotic domains in which our basic method is applicable, and conclude by providing an empirical evaluation of our algorithm in a robotic simulation.**

***Index Terms*—multi-robot systems, multi-robot formation, multi-robot task reallocation**

I. INTRODUCTION

This paper discusses a team of N robots engaged in a cooperative task. Specifically, we consider the problem of choosing k team members in order to reassign them to a new task. We assume that all members are capable of participating in the execution of the new task, and the cost of the new task does not depend on the identity of the robots chosen to perform it. Therefore this work concentrates on the problem of choosing k robots such that the performance of the existing task, performed by the remaining group members, will be as efficient as possible.

The general problem of choosing k out of N team members in order to perform a new task is an important problem [6], [9], [10], [13], [14], [18], [30]. Previous work in task reallocation in teams of agents usually concentrate on optimizing some group-objective function that corresponds to the abilities or characteristics of the agents (e.g. [11]). The general problem was proven to be $\mathcal{NP}$-hard as a special case of the Set Partitioning Problem [12].

In this work we suggest a new method for choosing team members for task reallocation that is based on the interaction between team members. Our approach concentrates on the minimization of the cost of interaction between the entities, rather than on the capabilities of the agents. Under this model, the problem remains $\mathcal{NP}$-hard, however we manage to substantially reduce the number of possibilities for optimal assignments, thus we reduce the search domain from order $\binom{N}{k}$ to order 2^k. This reduction makes it possible to solve the problem faster even using a brute-force algorithm, and in common cases where k is relatively small it is possible to solve the problem both optimally and efficiently (i.e., in polynomial time).

In the model we propose, the set of team members and the interaction between them is represented by a weighted directed graph, where the vertices represent the members, the directed edges represent the interaction (interaction of a vertex a to vertex b could be different from interaction of b to a, thus the edges are directed) and edge weights represent the cost of the interaction between them. We assume the team has a leader, which can correspond to a formation leader in the example accompanying us throughout the work. In the general case, this leader is a team member that has only incoming edges and no outgoing edges, i.e., it does not depend on other team member's input in its interaction with them.

The problem used to illustrate methods is the fundamental problem of maintaining a formation by a team of robots; this problem has received considerable attention in the literature (e.g. [2], [3], [16], [21]). In order to maintain the formation, a robot has to monitor one or more robots in the formation. Several common methods for choosing the identity of the robot(s) to be monitored are known [2]. We choose to focus on the method proposed in [16], in which the identity is driven by minimization of the monitoring cost of the robots. Therefore the graph is the monitoring graph which represents the sensorial capabilities of the robots, i.e., which robot can sense the other and at what cost. The problem thus involves extracting k robots in order to perform a new task (for example acquire a new target) while minimizing the monitoring cost of the remaining group.

In particular, in this work we make the following contributions.

1) We introduce a new method in which the problem of choosing k out of N team members is modeled by a graph, and the decision is taken while emphasizing the cost of interaction between the members.
2) We describe a deterministic algorithm for choosing k

Preliminary results appeared in Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI'05) [1]

This research is supported by NSF Grant #0705587 and ISF Grant #1685

Noa Agmon is with Department of Computer Science and Applied Mathematics. Weizmann Institute of Science, Israel.
E-mail: noa.agmon@weizmann.ac.il

Gal A Kaminka, Sarit Kraus (also affiliated with UMIACS) and Meytal Traub are with Computer Science Department. Bar Ilan University, Israel.
E-mail: {galk, sarit, traubm}@cs.biu.ac.il

team members while minimizing the cost of interaction between the members of the remaining group assuming that the group has one possible leader. The algorithm reduces the trivial time complexity of $\mathcal{O}(N^k)$ to $\mathcal{O}(2^k)$, thus is polynomial if we assume that $k = \mathcal{O}(\log N)$.

3) We demonstrate the generality of the basic algorithm by showing how it extends to deal with the following cases:
   a) The group has more than one member which can potentially act as the leader. These cases are significantly more complex, as they require that we check possible leader replacements; yet we show that they can still be done optimally by applying an algorithm that works in reduced time (exponential in k, hence polynomial for $k = \mathcal{O}(\log N)$).
   b) Not all team members can be chosen for the new task. We show that in some cases the basic algorithm can still be used under these circumstances.
   c) The cost function of the robots is composed of more than one component. In particular, we consider weighted components where we each component is associated with a relative weight, and prioritized components of the cost function, where the components are considered according to their priority.
4) We show that the method we propose, and the basic algorithm within it, is a general method that can be used in several other domains.

We then present an empirical evaluation of the algorithm, using the Player/Stage simulated environment [15]. This implementation illustrates the use of three different task reallocation algorithms—the basic algorithm, and two algorithms that take other considerations into account (weighted and prioritized components)—in different formations.

II. Related Work

Our problem, task reallocation in multi-robot formation, is on the line between two canonical problems in multi-robot systems, multi-robot pattern formation and maintenance [2], [8], [16], [17], and multi-robot task allocation [9], [10], [14], [20], [22], [28].

The only work, to our knowledge, that considers the problem of task allocation in multi-robot formation is the work by Michael et al. [19]. In their work, they use combinatorial auctions in order to assign n robots to m tasks, i.e., splitting the robots into m groups. They implement their proposed solution in a team of mobile robots in a formation under the constraint of collision avoidance. In our work we also provide an empirical evaluation of the case in which it is required to avoid collisions, however our main general method concentrates on optimize the cost of interaction (sensing) amongst the remaining team. Note that our work does not require communication, as the deterministic algorithm can be executed by each robot individually.

The class of pattern formation and maintenance of a team of mobile robots received considerable attention in the literature [2], [8], [16], [17]. Balch and Arkin [2] discussed the reactive behavior-based techniques for formation control, motivated by nature phenomena. They describe three possible basic techniques a robot can use in order to maintain its position in a formation: leader referenced, neighbor referenced and unit-center referenced.

Several works used the neighbor referenced technique, and examined it in different aspects. Desai [8] offers a graph modeling of a formation using *control graphs*, based on the neighbor referenced technique, where the robots can change their formation by switching between control graphs. Lemay et al. [17] consider the problem of initially assigning robots to specified locations in the formation. Kaminka et al. [16] assume that a robot follows not necessarily the closest neighbor, but the one that the cost of sensing it is minimal, and represent the formation in a sensing graph, i.e., a graph that represent the cost of sensing. We use the work of Kaminka et al. as baseline for our work, thus choose to reallocate k robots to a new task based on the sensing graph.

Balch and Hybinette [3] presented a new behavior based approach for formation control, based on potential functions that attract robots, in designated positions, from their team members in the formation. The advantage of this new approach is that it is easily scalable to large formations, it assumes only local sensing and it is flexible in the sense that it allows to work in various formations and avoid obstacles. This work concentrates on determining the proper formation position for each team member.

Our problem belongs to the ***multi robot task allocation*** (MRTA) domain. Following the taxonomy for MRTA systems given by Gerkey and Mataric [14], our work deals with instantaneous assignments of single-task robots performing multi-robot tasks.

Parker [20] presents a behavior based architecture ALLIANCE for cooperative multi-robot control. This architecture deals with missions that are composed of loosely coupled subtasks that are independent. The task achieving behaviors are divided into lower-lever behaviors that correspond to individual ad-hoc primitives such as obstacle avoidance, and high-level behaviors that corresponds to the team-mission actions. This work, however, does not deal with task reallocation within the team of robots but in the individual behavior of the robot that assures completion of team tasks.

Studies that discuss the problem of choosing k out of N agents in order to perform a new task mostly concentrate on maximizing the profit gained by the optimal performance of the new task. For example in [9], [10] Dias et al. consider market-based coordination methods for assigning tasks to robots. In such systems, tasks are allocated between robots using auctions. Finding the optimal assignment using combinatorial auctions, where bidders can bid on combinations of items, has been proven to be $\mathcal{NP}$-hard [23]. We, on the other hand, concentrate on maximizing the benefit (minimizing the cost) gained by the optimal execution of the original task. The problem can be considered equivalent, if it is perceived as choosing $N-k$ team members to perform the original task. Nevertheless, we aim to find the exact optimal solution where in other studies the problem is handled heuristically.

Other studies discuss allocation of agents to several given tasks. These studies mostly provide heuristic algorithms for

efficient allocation of agents to tasks. In [22], Sander et al. describe task allocation heuristic algorithms for settings in which the agents and tasks are geographically dispersed in the plane.

The problem of designing and forming groups of agents while maximizing some mutual objective is usually referred to as coalition formation. The work of Shehory and Kraus [28] is close to the scenario discussed in this paper. They suggest algorithms for coalition formation among agents, where an agent can be either a member of only one coalition (similar to our case), or coalitions may overlap. They provide heuristic algorithms, rather than an exact optimal solution.

Sandholm and Lesser [25] also dealt with coalition formation, but with self interested agents. Our problem can be perceived as a private case of the general coalition formation studies (forming two coalitions - one for the old task and one for the new).

Tosic and Agha describe a distributed algorithm for generating coalitions based on the current physical configuration of the agents, using maximal cliques [29]. They show that the agents convert to the same coalitions, but their work does not refer to any kind of group utility, as opposed to our work, which maximizes the joint utility of the agents performing the original task.

Another subject in coalition formation which is closely related to the problem discussed here is the problem of coalition structure generation [7], [24], [26]. In this problem, which has been shown to be $\mathcal{NP}$-hard, a division of the agents into coalitions is searched such that the group utility (payoff) is maximized. Sandholm et al. [24] and Dung and Jennings [7] provide algorithms for coalition formation with a guaranteed lower bound from the optimal solution. Sen and Dutta describe a stochastic search approach for searching for optimal coalitions [26]. As opposed to our scenario, whereby an optimal solution is guaranteed, in these studies a sub optimal solution is given.

III. Problem definition

The motivation for the following representation of the problem comes from the world of multi robot formation maintenance. In this domain, the robots monitor one another in order to maintain a formation. While in the formation, k of the robots are required to leave the team in order to acquire a new target. We wish to extract the robots in such a way that will minimize the disruption to the original formation. A detailed description of the multi robot problem follows the general definition of the problem, which is applicable to other domains (as seen in Section VII).

Let $G = \{P_1, \ldots, P_N\}$ be a group of N homogenous team members. The interaction between team members can be represented by a cost function. Note that this cost function can be generalized into an interaction-based *utility* function. The group has one root - a team member that acts as the leader. The set of members and the interaction between them can be presented as a directed weighted graph $G = (V, E)$, where $v \in V$ are the team members, and the edges represent the interactions. Namely, $(v_i, v_j) \in E$ if an interaction exists between v_i and v_j with a $\text{cost}(v_i, v_j)$, which is the weight of the edge (v_i, v_j). An *Optimal Interaction Tree* (OIT) is built on this graph by simply running Dijkstra's shortest path algorithm between all vertices of the graph and the leader (similar to [16]).

One main constraint is required on this graph as follows:

Constraint A: If $(v_1, v_2) \in E(\mathsf{OIT})$ and $(v_2, v_3) \in E(\mathsf{OIT})$, then $\text{cost}(v_1, v_2) + \text{cost}(v_2, v_3) < \text{cost}(v_1, v_3)$.

This means that the triangle inequality exists on the *optimal* tree of G, $\mathsf{OIT}(G)$ (but not necessarily on G). Intuitively, if a path is chosen to be in $\mathsf{OIT}(G)$, then there is no shorter path between the two edges of the path. Note that this does not mean that the triangle inequality holds for general edges of G, as depicted in Figure 1. In this example, the triangle inequality does not hold in G, for example $\text{cost}(a, c) + \text{cost}(c, e) \leq \text{cost}(a, e)$, thus clearly the edge (a, e) will not be in $\mathsf{OIT}(G)$. The triangle inequality can still exist between some vertices, for example $\text{cost}(a, b) + \text{cost}(b, c) > \text{cost}(a, c)$.

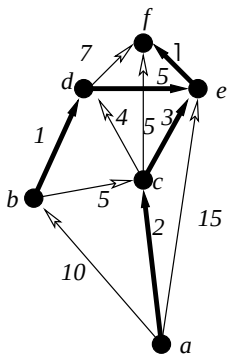

Fig. 1. An example of a case in which the triangle inequality exists in $\mathsf{OIT}(G)$, yet it does not exist in G.

The motivation behind this constraint is as follows. In order to achieve a relatively lower time complexity, we would like to restrict the options of the removal of nodes. Specifically, this constraint allows us to consider only leaves or subtrees of the tree. The constraint is necessary in order to avoid cases like the one depicted in Figure 2, in which the structure of G does not follow Constraint A. In this case, illustrated in our robotic domain, the cost of the sensing robot r_1 by r_3 is 7, and lower than the cost of sensing r_1 by r_3 via robot r_2 ($10 + 10 = 20$). However, the edge (r_3, r_1) does not appear in the graph since r_2 lies exactly between r_3 and r_1, thereby blocking the sensing capabilities of r_3. This example conflicts with constraint A since edge (r_1, r_3) is not in the tree, whereas $\text{cost}(r_3, r_1) < \text{cost}(r_3, r_2) + \text{cost}(r_2, r_1)$, and indeed it would have been more profitable to remove r_2 and not r_3 which is the leaf.

In many cases the leader of the team has unique characteristics that distinguish it from the other team members. Consequently there is no point in discussing the removal of the team leader. However, in some cases all robots are homogenous, including the leader, thus an additional property can be added to the problem, as given in the following

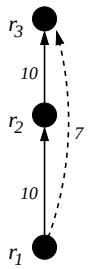

Fig. 2. An example of a case where Constraint A is not fulfilled.

definition.

Definition 1: In a graph $G = (V, E), V = \{v_1, \ldots, v_N\}$, a *potential leader* group $L = \{\tilde{v}_1, \ldots, \tilde{v}_M\}$, $1 \leq M \leq N$ is a subset of V containing possible leaders from the group. We denote the size of the potential leader group by M.

Having the OIT of the group, $k < N$ vertices should be extracted from the graph. The extraction should be done while satisfying the following basic objectives.

1) The cost of the remaining OIT should be minimal.
2) At least one of the vertices from the potential leader group of G should remain in the graph (this requirement is necessary due to the fact that we assume the remaining formation must have a leader, and only one of the potential leader group members can act as a leader).

It is assumed that all N team members can theoretically be extracted, hence if we are dealing with acquiring a new target or performing a new task, then all members are compatible for the mission (see an exception to this assumption in Section IV-C). Therefore, potentially, the number of different possibilities for extracting the k team members from the group is $\binom{N}{k}$. The algorithms described later in this paper, which work in our settings and under our constraints substantially reduce the complexity to $\mathcal{O}(2^{k/2})$.

Returning to the multi-robot domain, the root of the tree is the formation leader, the original graph is the monitoring multigraph where each vertex represents the location of a robot, and the edges represent the monitoring capabilities and cost of each robot of its peers. As proposed by Kaminka et al. in [16], an Optimal Monitoring Tree, OMT, which is a special case of the OIT, is built on the monitoring multigraph. The OMT describes who each entity should monitor in order to minimize the cost of the sensing path from itself to the leader. The value of monitoring multigraphs and specifically OMTs, is its compatibility with real world scenarios, i.e., in the real world robots usually have limited sensing capabilities and the cost of sensing varies from one sensed object to another—depending on its distance and angle with respect to the sensing robot. When extracting k robots, the objective is to minimize the cost of sensing of the remaining group.

IV. Reallocation while minimizing costs

In this section we describe three variants of the team reallocation algorithm. In the first and basic algorithm, k robots are extracted from a team of N robots, where the formation has one specified leader that cannot be extracted. In the second algorithm, the formation has a set of possible leaders, where the only restriction in the extraction is that one of the possible leaders will remain in the formation as the leader. In the final variant of the algorithm described in this section we discuss the case in which some robots cannot be extracted from the formation.

A. Team member reallocation with one possible leader

In this section we describe an algorithm that finds the optimal k vertices that should be extracted from the graph such that the cost of the remaining OIT is minimized. The Tree_Pruning algorithm described in this section, finds the optimal k vertices to be extracted, assuming that the leader cannot be changed.

The following lemma and its corollary provides the motivation behind the algorithm. The lemma proves that the algorithm should concentrate on removing the leaves or subtrees from the OIT rather than arbitrary vertices without all vertices that are in the subtree rooted in that vertex. The reason lies in the fact that the OIT is built in an optimal manner, hence any removal of a vertex without all the subtrees rooted in it will force the vertices of those subtrees to find an alternative path towards the leader. Based on the optimality of the tree, this alternative either will incur the same cost or will be more expensive than the original one.

Lemma 1. *Consider an* OIT(G)*, satisfying Constraint A. If vertex v that is not a leaf nor the leader is removed, then the sum of the weights of the edges of* OIT$(G \setminus v)$ *will not decrease.*

Proof: In a DAG, every vertex that is not a leaf is an articulation vertex, meaning, removing it will disconnect the graph. Therefore all vertices connected to v should find another node to connect to, i.e., all $u_i \in V$ such that $(u_i, v) \in E$ and $(v, u) \in E$, should create a new edge (u_i, v_j) such that the cost of the DAG is minimized.

If $v_j = u$ then the proof is completed, since by Constraint A $\text{cost}(u_i, u) > \text{cost}(u_i, v) + \text{cost}(v, u)$. If $v_j \neq u$ then by minimality of the OIT it follows that if $\text{cost}(u_i, v_j) < \text{cost}(u_i, v)$ then the algorithm would have chosen u_i to point to v_j in the first place, contradicting the minimality of the OIT. ■

Corollary 2. *In an* OIT(G) *satisfying Constraint A, the benefit gained from removing a leaf is greater than the benefit gained from removing one of its ancestors.*

The following definitions are used throughout the description of the algorithm.

Definition 2:

2.1 A *palindromic composition* of a number k is a collection of one or more positive integers whose sum is k. The number of palindromic compositions of a number k is $2^{\frac{k}{2}}$ [4].

2.2 A *bundle* originating in vertex v is the subtree rooted in vertex v. Vertex v *nests a bundle* of size t if the bundle originating in it has t vertices, including v.

2.3 In a directed tree $G = (V, E)$ where all paths are directed to the root of the tree, if $(v, u) \in E$ then u is called v's *pivot*.

From Corollary 2 we can assert that the gain from removing bundles or leaves from OIT(G) will be greater than the gain from removing arbitrary vertices from the graph. This fact motivates the algorithm Tree_Pruning, which exhaustively searches all possible combinations of leaves and bundles and picks the combination which results in the highest reduction in cost to the remaining OIT(G).

The Tree_Pruning algorithm (1) works as follows. First, it creates a table in which it stores the vertices in levels $1, \ldots, k$, where each level i, $1 \leq i \leq k$, contains vertices that nest a bundle of size i. For each of these elements it indicates the gain from removing the bundle originating in that vertex. This gain is simply the sum of all the costs of edges in this subtree, including the cost of the edge going from the root of the subtree to its pivot. After the table is created, the algorithm starts checking all palindromic compositions of the number k. For each composition $\alpha_1 + \alpha_2 + \ldots + \alpha_t$ the algorithm first checks whether it is feasible, i.e., whether there are components of sizes $\alpha_1, \ldots, \alpha_k$. If so - for each α_i it checks for the element with maximal gain in level α_i of the table. If the algorithm picks up non-disjoint bundles, then it checks the gain of removing each element of the non-disjoint set alone. Summing up the gains from removing the composition is compared to the current maximal gain, and the set resulting in the higher gain is saved. Finally, after all palindromic compositions of k are examined, the algorithm returns the set with the highest cost reduction upon its removal.

Algorithm 1 Algorithm Team$_k$ = Tree_Pruning$(G = (V, E), k)$

1: **for** each leaf $v \in V$ **do**
2: Start building a k-bundle bottom-up:
3: $C_{best} \leftarrow 0$ and Team$_k \leftarrow \emptyset$.
4: Add each subtree of size $1 \leq t \leq k$ to the table in row t and calculate its cost.
5: Sort all elements in each row according to their cost.
6: Generate a palindromic composition of the number k and sort each composition from left to right in decreasing order.
7: **for** each possible composition C_j of $k = \alpha_1 + \alpha_2 + \ldots + \alpha_t$ **do**
8: Check whether the composition is feasible.
9: **for** each α_i in the composition, $i = 1, 2, \ldots, t$ **do**
10: Pick highest order unmarked element from row t_i and mark it.
11: **if** the elements are not disjoint, then check all possibilities: **then**
12: Pick element with highest α_i, next don't pick it and pick element with next highest α_i and so on.
13: Calculate the cost of the composition C_j.
14: **if** $\text{cost}(C_j) \geq C_{best}$ **then**
15: $C_{best} \leftarrow C_j$ and Team$_k \leftarrow$ current composition.
16: Return Team$_k$

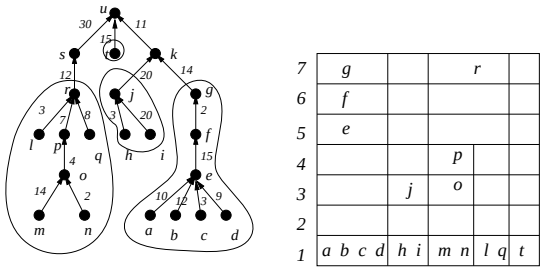

Fig. 3. An example of 7-bundling of a tree.

Theorem 3. *The* Tree_Pruning *algorithm finds the optimal k vertices to be removed from the group such that the cost of the remaining* OIT *is minimized.*

Proof: As seen in Corollary 2, the optimal benefit for the remaining tree is obtained by removing vertices that are not articulation points in the graph. Therefore the examination of all removal possibilities of leaves and bundles, as done by the algorithm, assures that the optimal group of k vertices will indeed be removed. ■

Time Complexity: The time complexity of the preliminary work of building the table is $\mathcal{O}(N)$, as each vertex is visited once. The sorting of the rows will cost additional $\mathcal{O}(N \log N)$ time. The number of palindromic compositions of a number k is $2^{\frac{k}{2}}$ [4]. The algorithm might check each composition (the worst case) N times, if the chosen elements are not disjoint. Assuming that each approach to an element takes $\mathcal{O}(1)$ steps (depending on the data structure used), each composition is calculated in (the worst case) $\mathcal{O}(N)$ steps.

Therefore the total time complexity of the algorithm is $N \log N + N 2^{\frac{k}{2}}$. Assuming that k is in order $\log N$, then the time complexity of the algorithm is $\mathcal{O}(N \log N + N\sqrt{N}) = \mathcal{O}(N^{1.5})$.

B. Team member reallocation with multiple possible leaders

Removing the leader v_{lead} can significantly decrease the total cost of the graph in cases where the weights of edges entering v_{lead} are considerably higher than the other weights in the graph. Therefore when examining the vertices, it can be highly profitable to examine removal of both leaves (or bundles) and the leader. The removal of the leader is possible only in cases where the potential leader group of the formation is of a size greater than one. Consider the case in which robots move in a formation. In this case, it is possible that several robots in the formation can lead the group. This is obvious in cases in which all robots are homogenous, whereby, theoretically, the number of robots in the potential leader group is N.

The order of extraction of the leading vertex is important, since changing the leader might result in a different structure of the OIT. In particular, it might result in changing the identity of the leaves and bundles. If we wish, for example, to extract three vertices from the graph, the result may vary if we pick a leaf, leader and a leaf from the new tree, as opposed to picking, say, two leaves and then the leader. Note that in some cases removal of a leaf might result in changing the leader,

depending on the leader election algorithm. However, in our case we assume that this does not happen.

In the extreme case in which $M = N$, the number of different possibilities for choosing k vertices - a combination of leaves and leaders, is $\mathcal{O}(2^k)$. See Figure 4 for an example.

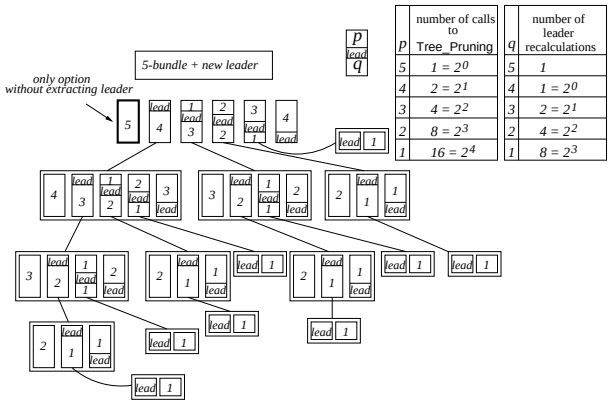

Fig. 4. An example of the search tree of all possibilities of extracting $k = 5$ vertices from the graph where the leader can be elected as well in each step.

In the first level, $l = 1$, we can either pick k vertices using the Tree_Pruning algorithm (meaning, we do not change the leader), or change the leader first, second and so on until the k'th vertex is selected. Each of these options except the former branches out similarly in the next level ($l = 2$) where k decreases by one. If $M \geq k - 1$ then the formal time complexity analysis is as follows.

Assume that a leader is chosen in a round where t vertices remain to be chosen, $1 \leq t \leq k$. It is possible to pick p vertices before the leader is replaced and q afterwards, $0 \leq p, q \leq t - 1$ and $p + q = t - 1$. Therefore there are t different choices of the order in which the leader is extracted, in addition to the choice where the leader is not replaced. When a leader is replaced, the calculation of the p vertices chosen prior to it is obtained simply by running Tree_Pruning for p. The remaining q launch an additional level where, again, they branch into $q + 1$ new options. In order to calculate the complexity of finding the best allocation, we need to calculate the number of times each p appears (the q is calculated in the next level). As demonstrated in the table below, each number $i = 1, \ldots, k$ appears as p, i.e., above the leader line, 2^{k-i} times. Using Tree_Pruning, the complexity of each extraction of size i is $N \log N + N2^{\frac{i}{2}}$. Therefore the total complexity is $\sum_{i=1}^{k} 2^{k-i} \cdot (N \log N + 2^{\frac{i}{2}}) = \mathcal{O}(2^k N \log N) + N \sum_{i=1}^{k} 2^{k-\frac{i}{2}} = \mathcal{O}(2^k N \log N) + \mathcal{O}(2^k N) = \mathcal{O}(2^k N \log N)$. If $k = \mathcal{O}(\log N)$, then the complexity of choosing the members is $\mathcal{O}(N^2 \log N)$.

To this complexity we need to add the cost of recalculating the graph after a leader is extracted. As shown in [5], the complexity of calculating the OIT of a graph of size N given the identity of the leader vertex is simply running Dijkstra's shortest paths algorithm that takes $\mathcal{O}(N^3)$. If there are $\tilde{M}$ potential leaders, then the complexity is $\mathcal{O}(N^3 M)$. Hence here the time complexity can be bounded from above by the total number of qs, times $\mathcal{O}(N^3 M)$. The total number of qs, as demonstrated in Figure 4, is $\sum_{i=1}^{k-1} 2^{(k-1)-i}$, therefore the total time complexity is $N^3 M \sum_{i=1}^{k-1} 2^{(k-1)-i} = \mathcal{O}(N^3 M 2^k)$. Again, in our case $k = \mathcal{O}(\log N)$, hence the final complexity of the algorithm is $\mathcal{O}(N^4 M)$.

C. Addressing non-removable robots

Sometimes, there may be a requirement that some team members must not be extracted from the group. For example, if the robots moving in the formation are heterogenous, some might be incapable of performing the new task and thus cannot be chosen to do it. In another example, if the formation itself defines critical positions that cannot be deserted, it will dictate the identity of the robots that cannot be extracted. In both cases—driven by the old or new task requirements—some robots must remain in the original team. Converting it to our graph problem, if vertices should be extracted only from a subgroup G_1 of G, $G_1 \subseteq G$ and $|G_1| \geq k$, then a small variation of the basic algorithm can be used in some cases. First, if $|G_1|$ is exactly k, then the only option is that all vertices in G_1 be extracted.

Definition 3: In an OIT graph G, a vertex $v \in V(G)$ is called a *bundle blocker* if it cannot be extracted from the graph and the bundle above it stops spreading.

In our case, the bundle blockers are all vertices u_i such that $u_i \notin G_1$. If the bundle blockers lie in accumulating levels of depth k or more, then a simple variation of Tree_Pruning can be used in order to find the optimal k vertices to be extracted. In this variation of Tree_Pruning, the algorithm should be run on the OIT graph G, but should stop at bundle blockers or at depth k, whichever comes first.

V. Multiple components in the cost function

When establishing the nature of the cost function, it is possible that more than one property will be taken into consideration. For example, one might want to consider both the cost of the remaining formation and the cost of the new formation. We consider two manners in which more than one consideration can be incorporated in the cost function: prioritized components and weighted components. We show examples in which the Tree_Pruning algorithm can be used in both cases.

Note that the ground rule which stands at the base of the use of the Tree_Pruning algorithm is that it is more profitable to remove leaves and bundles of the OIT. Hence any scenario in which we can incorporate the use of Tree_Pruning must apply to this rule. In the following examples we were motivated by the stability of the formation, whereby any change in the pivot of robots might cause instability of the formation. Therefore the following case suits the requirement that only bundles or leaves be removed, and thus the Tree_Pruning algorithm perfectly suits this problem.

A. Prioritized cost components

In prioritized cost components, the components are presented in a prioritized list, namely, component one is more important than component two and so on. Therefore we first minimize the cost according to the first priority. In case of a

tie, we examine the component listed second in the priorities, and if that results in a tie we move on to the third priority component and so on.

A good example of this would be when we wish to minimize events that might undermine the remaining group's formation stability as well as minimize the monitoring cost. For instance, we assume that robots leaving the formation in order to acquire a new target continue in a straight line from their current location towards the new target. First, we would like to cause minimal changes to the current OIT, so that robots will not have to switch pivots and thereby cause temporary instability. Thus only leaves or bundles should be removed, and the leader should remain intact. Second, we wish to minimize collisions between the robots leaving for the new target and the ones remaining in the formation. As seen in Figure 5, if v_2 moves straight towards t_G and maintains its predefined velocity, it will intersect with v_3. In addition, if v_5 moves towards t_G then at some point it will block v_4's view of its pivot, v_3. Third, we wish to minimize the monitoring cost of the remaining formation. Hence the prioritized components list is as follows.

1) Minimize collisions between the robots leaving for the new target and the ones remaining in the formation
2) Minimize the incidents of robots leaving the formation while, at some point, crossing an OIT edge, thus hiding the pivot of some robot remaining in the formation and potentially causing it to divert from the group formation.
3) Minimize the remaining group's monitoring cost.

The Prioritized_Pruning algorithm works as follows. First, assuming that the robots are homogeneous, it is simple to calculate the expected intersections between paths of robots leaving the formation and the remaining OIT vertices (robots) and edges. For each robot r_i (vertex v_i) that leaves towards goal point p_G, the algorithm checks against all robots $r_j \neq r_i, r_j \neq r_{lead}$ whether the path of r_i hides the outgoing edge from robot r_j (vertex v_j). If so, it adds t_j to the entry of r_i in a pre-specified Table under column E (see for example Figure 5). If the robots themselves intersect, then r_j is added to Table under column I. After creating this table, Tree_Pruning is run on the graph where three features are examined in each step: I intersections, E intersections which are extracted from the Table, and the cost incurred by the remaining OIT (in this order).

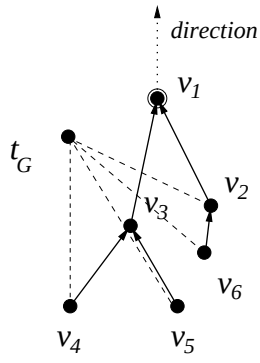

Table:

	E intersection	*I intersection*
v_2	Ø	v_3
v_3	Ø	Ø
v_4	Ø	Ø
v_5	v_4	Ø
v_6	v_4 v_5	Ø

Fig. 5. An example of a path/edge intersection.

The Prioritized_Pruning algorithm is guaranteed to find the k robots that will minimize the potential disturbance to the formation satisfying the criterions we defined. The time complexity of the algorithm is composed of two steps. In stage 1, a simple brute force algorithm that finds the intersections will take $\mathcal{O}(N^2)$ steps by simply comparing each pair of robots. Stage 2 is similar to the Tree_Pruning algorithm with an addition of maximum $\mathcal{O}(k)$ comparisons at each step, hence the complexity is $\mathcal{O}(N^{1.5}\log N)$ (assuming that $k = \mathcal{O}(\log N)$), and altogether the complexity is $\mathcal{O}(N^2)$.

Algorithm 2 Algorithm $\mathsf{Team}_k = \mathsf{Prioritized\_Pruning}(G = (V, E), k, t_G)$

for each vertex $v_i \in V$ such that v_i is not the leader **do**
  Go over all vertices of the graph except for vertices in the bundle originating in v_i.
  if the outgoing edge of vertex v_j intersects the path of v_i on its course towards t_G **then**
    Add v_j to $\mathsf{Table}(v_i, E)$.
  if v_i on its course towards t_G intersects v_j **then**
    Add v_j to $\mathsf{Table}(v_i, I)$.
Run procedure $\mathsf{Tree\_Pruning}(G, k)$ with the following modifications.
Set $E_{best} \leftarrow \infty$, $I_{best} \leftarrow \infty$ and $\mathsf{Team}_k \leftarrow \emptyset$.
Check the number of E and I intersections between members of the current chosen elements and the remaining ones and store them in E_{cur} and I_{cur}, respectively.
if $I_{cur} < I_{best}$ **then**
  $I_{best} \leftarrow I_{cur}$ and
  $\mathsf{Team}_k \leftarrow$ current composition.
if $I_{cur} = I_{best}$ and $E_{cur} < E_{best}$ **then**
  $I_{best} \leftarrow I_{cur}$ and $\mathsf{Team}_k \leftarrow$ current composition.
if $I_{cur} = I_{best}$ and $E_{cur} = E_{best}$ **then**
  Check the difference between the cost of the composition and save the best of two choices, as done in Tree_Pruning .
Return Team_k.

B. Weighted cost components

Formally, weighted cost components allow us to assign an accumulating percentage for each component, and choose the option resulting in the minimized cost value U. Each of these cost values is composed of l different components $\{u_1, \ldots, u_l\}$, and w_t is the weight of the cost component u_t, where $\sum_{t=1}^{l} w_t = 1$. Therefore the weighted cost value U is calculated as: $U = \sum_{t=1}^{l} w_t^i u_t^i$

In the following example we assume that robots leaving the formation remain in their current location, i.e., they will change their relative position to other robots. Our objective is to minimize both the remaining group's monitoring cost and the monitoring cost of the robots leaving the formation, according to their location at the moment they leave the formation. This is applicable in cases where a subgroup of robots is required to leave the formation and create a formation of its own with minimal sensorial cost of the new formation. Our original assumption is that we wish to minimize disruptions of the original formation, thus we will examine the removal of only leaves and bundles, and leave the leader

intact. This property will allow us to use the Tree_Pruning algorithm in this case. Note that if we consider only the monitoring cost of the leaving robots, we might have needed to remove vertices that are not necessarily leaves or bundles. For example, if $k = N - 2$, then the optimal choice of removal would be the removal of the minimal edge in the OIT, regardless of its location if only the monitoring cost of the extracted group is considered. Therefore, as we have shown in the previous subsection, the initial assumption that we remove only leaves and bundles is crucial for the use of the Tree_Pruning algorithm and ensures its complexity.

Minimizing the remaining group's sensorial cost (first component) contradicts, in most cases, the minimization of the extracted group's sensorial cost. This contradiction is exemplified in Figure 6, and results straightforward from the fact that the first minimization requirement would cause the removal of the most expensive edges, while the second component would require the removal of the least expensive edges from the graph.

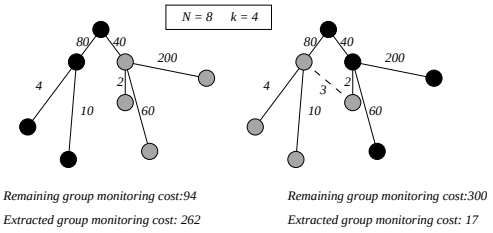

Fig. 6. An example of a contradiction between the two cost components: the remaining group's OIT cost and the extracted group's OIT cost. Here $N = 8, k = 4$ and the optimal choice of nodes for removal is colored in gray, while the remaining nodes are colored in black.

Use w_1 to denote the weight of the cost component of the remaining formation's monitoring cost , and w_2 to denote the weight of the cost component of the extracted formation's monitoring cost. The Weighted_Pruning algorithm, then, works as follows. For each possible choice of k robots, the algorithm calculates the cost of the remaining OIT multiplied by w_1, the cost of the extracted OIT multiplied by w_2, and it sums the two values. If the resulting value is lower than the lowest value obtained so far, this is saved as the potential optimal choice. After all choices have been checked, the choice with the optimal cost is reported.

Algorithm 3 Algorithm $\mathsf{Team}_k = \mathsf{Weighted\_Pruning}(G = (V, E), k, w_1, w_2)$

1: Run procedure $\mathsf{Tree\_Pruning}(G, k)$ with the following modifications.
2: Set $E_{best} \leftarrow \infty$, $I_{best} \leftarrow \infty$ and $\mathsf{Team}_k \leftarrow \emptyset$.
3: $C_j \leftarrow$ current composition.
4: $C_R \leftarrow$ cost of $\mathsf{OIT}(G \setminus C_j)$, and $C_E \leftarrow$ cost of $\mathsf{OIT}(C_j)$
5: $C_{cur} \leftarrow w_1 \times C_R + w_2 \times C_E$.
6: **if** $C_{cur} < C_{best}$ **then**
7: $C_{best} \leftarrow C_{cur}$ and $\mathsf{Team}_k \leftarrow C_j$.
8: Return Team_k.

Algorithm Weighted_Pruning is guaranteed to find the k robots that will minimize the total cost according to the weights we received from the user. The time complexity of the algorithm is identical to the time complexity of the Tree_Pruning algorithm, since we go over the entire graph only once for each possible composition,, which leaves us with a time complexity of $N2^{\frac{k}{2}}$ needed to check all compositions.

VI. Empirical Evaluation

We implemented the three algorithms described herein, Tree_Pruning, Prioritized_Pruning and Weighted_Pruning, in order to perform an empirical evaluation of the algorithms. The implementation was done using the Player/Stage simulation package [15], a practical and popular development tool for both simulated and real robots.

We simulated 16 robots, traveling in one of three formations commonly tested in general multi-robot formation problems (e.g. [16]) - see Figure 7: *A.* Diamond *B.* Triangle *C.* Arrowhead. The edges between the robots in each formation were given weights according to the cost of sensing, similar to the weights given in [16]. In the first step, the robots built a spanning tree, instructing each robot which other robot to monitor in order to minimize the cost of sensing inside the formation. The Spanning trees are indicated by bold arcs in Figure 7.

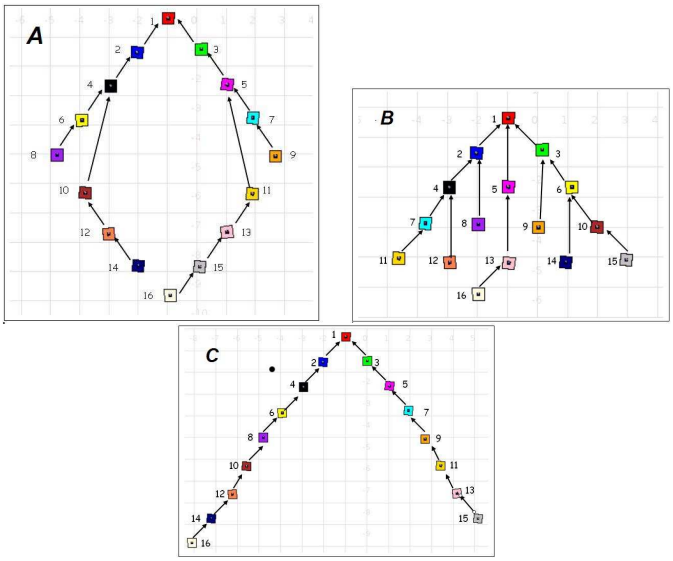

Fig. 7. Three different formations tested in our Player/Stage simulation. The arcs represent the edges of the minimal sensing tree from all robots to the leader (robot 1).

Following the first phase of constructing the minimal spanning tree, we executed the three algorithms on the formation: Tree_Pruning, Prioritized_Pruning and Weighted_Pruning. We were interested in revealing which robots were extracted as the output of each algorithm, or more specifically what different outputs would be returned by each algorithm for the same formation. We wanted to test whether the Prioritized_Pruning algorithm would indeed answer possible problems raised by the use of Tree_Pruning in the multi-robot formation domain. We continue with a

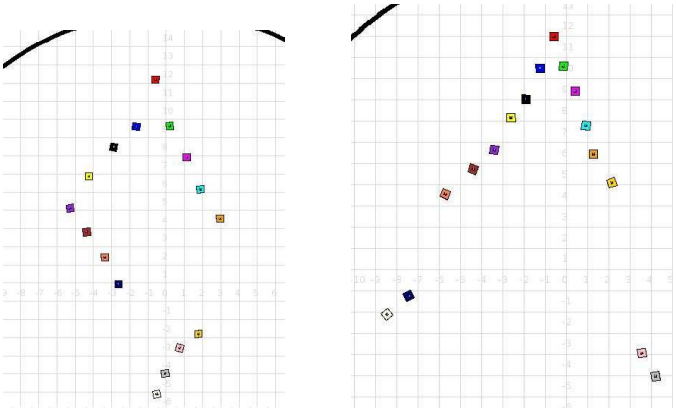

Fig. 8. Execution of the Tree_Pruning algorithm on formation A (left) and C (right). The snapshot was taken approximately 15 seconds after the extraction.

description of the results of the algorithms' performance on the chosen formations.

First, the robots executed Tree_Pruning in order to reallocate 4 out of the 16 team members to a new task. The extracted robots were instructed to remain in their position while the remaining formation continued their movement in their initial direction. We obtained the following results. In formation A, robots $11, 13, 15$ and 16 were extracted from the team (Figure 8). In formation C, robots $13, 14, 15$ and 16 were extracted (Figure 8). The more interesting results were obtained in formation B, where robots $8, 9, 12$ and 14 were extracted. Since the extracted robots remained in place and the remaining formation continued straight, robot 8 collided with robot 16 (see Figure 9).

The choice of robots to be extracted in formation B using Tree_Pruning strengthens the motivation to use the Prioritized_Pruning algorithm. Indeed, when executing the Prioritized_Pruning algorithm for formation B, the extracted set of robots was $9, 11, 14$ and 15. In this case, the target point of the robots was behind the formation, simulating a similar behavior given to algorithm Tree_Pruning, and thus emphasizing how the Prioritized_Pruning algorithm is able to solve the problem evolving from the use of the Tree_Pruning algorithm (see Figure 9).

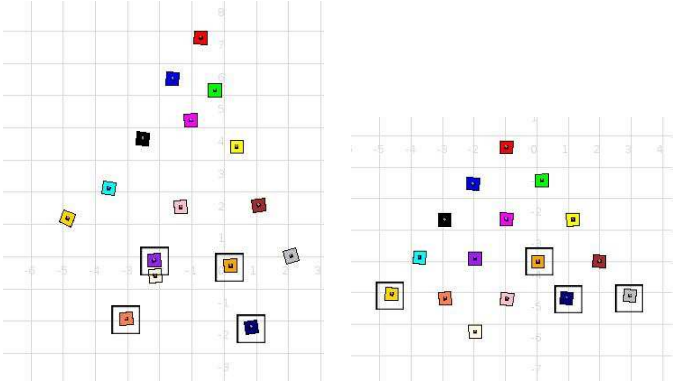

Fig. 9. Execution of the Tree_Pruning (left) and the Prioritized_Pruning (right) algorithms on formation B. The extracted robots are denoted by a surrounding square.

When we used the Weighted_Pruning algorithm with $w = 0.5$, the extracted robots were $11, 13, 15$ and 16 in formation A, which is the same set that was extracted by Tree_Pruning. However, in formations B and C when adding the consideration of the weight of the extracted team of robots, the set of extracted robots was different. In formation B the set of extracted robots was $4, 7, 11, 12$, and in formation C the set was $10, 12, 14$ and 16. We checked the output also given other weights ($w = 0.2$ and $w = 0.8$), however in both formations the resulted chosen team was similar. This can be explained by the fact that the weight of edges between the robots that cannot sense one another is ∞. Consequently any choice of a set of robots that would be extracted such that even one robot would remain disconnected from the other team members would receive a weight of ∞, and therefore that set would not chosen.

VII. Applicability in additional domains

The general representation of the problem as *team member* reallocation, makes it applicable in other domains, in addition to the multi-robot formation domain, which motivated the current study.

One example is a variation of the ***dependency tree*** [27]. A dependency tree $G = (V, E)$ describes a group of N tasks (vertices) with prerequisite relation, i.e., an edge $(u, v) \in E$ exists if u has to be executed before v, and $\mathrm{cost}(u, v)$ is the cost of executing v after u. The root of the tree is, then, the task that has to be executed last. In our case, we use a slight variation of the dependency tree. Here, we are given one task that should be conducted last and the interaction between all other tasks. If two tasks v_1 and v_2 are independent, then if v_1 is executed before or after v_2 their cost will be the same. If v_1 and v_2 are dependent, then without loss of generality, v_1 can rely on the fact that v_2 will perform a part of its task, thus $\mathrm{cost}(v_1, v_2)$ in this case will be smaller than the cost in the independent case. The OIT describes the optimal tree of execution of the tasks. The requirement is to remove k tasks from the group such that the cost of the remaining execution tree is minimized.

The ***warehouse assembling*** problem presents an additional example in which the OIT is applicable. In the warehouse assembling problem we are given a set of N warehouses located in N distinct positions, and all the trucks are heading towards one main warehouse. The vertices of the graph represent the warehouses, and the edges represent the distances between two warehouses (note that triangle inequality does not apply). The objective is for any number of trucks to visit all warehouses in minimal time. The OIT represents the optimal tree of paths from all warehouses to the main warehouse. The number of trucks t is, then, the number of leaves in the OIT. The requirement is to close k warehouses in order to cut back expenses while not increasing the value t, and thus remain with the assembling tree with the lowest cost.

The last problem is the ***network broadcast*** problem, in which we are given a network with one source vertex that should constantly broadcast messages to the rest of the network. The edges represent the cost of the link between every

two vertices, and the OIT is the optimal broadcast tree. Once again in this case we are required to remove k vertices in order to cut back expenses and remain with a broadcast tree with the lowest cost.

VIII. Conclusions

We considered the task reallocation problem in multi-robot formation. In this problem, a team of N robots move in a formation, and k of them need to be extracted from the group. The extraction is done considering the interaction cost between the team members—in our case the cost of sensing inside the formation—where the goal is to minimize the interaction cost between the remaining team members (and thus maximizing the utility function of the remaining group). A summary of our contributions in this paper is as follows.

- We introduce a new method in which the problem of reallocating k out of N team members to a new task is modeled by a graph, and the utility function is based on the interaction cost between the team members.
- We describe a deterministic algorithm for the reallocation problem which reduces the time complexity of the solution from $\mathcal{O}(N^k)$ to $\mathcal{O}(2^k)$. This result is shown for both cases in which the formation can have either one or more possible leaders.
- We generalize the use of the basic reallocation algorithm for cases in which the cost function has more than one component. In particular, we consider weighted components and prioritized components of the cost function.
- We describe an empirical evaluation of the algorithm and its variations using the Player/Stage simulated environment.
- We show that the method we propose that focuses on the interaction between team members, and the basic algorithm within it, is a general method that can be used in several other domains different from the multi-robot formation domain.

There are several areas we plan to pursue in our future work, which include considering the cost/utility of the robots leaving the formation and uncertainty in the actual cost of interaction.

References

[1] N. Agmon, G. A. Kaminka, and S. Kraus. Team member-reallocation via tree pruning. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, pages 35–40, 2005.

[2] T. Balch and R. Arkin. Behavior based formation control for mutlirobot systems. *IEEE Transactions on Robotics and Automation*, 14(12):926 – 939, 1998.

[3] T. Balch and M. Hybinette. Social potentials for scalable multi-robot formations. In *IEEE International Conference on Robotics and Automation (ICRA '00)*, volume 1, pages 73 – 80, 2000.

[4] P. Chinn, R. Grimaldi, and S. Heubach. The frequency of summands of a particular size in palindromic compositions. *Ars Comb.*, 69, 2003.

[5] T.H. Corman, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[6] T. S. Dahl, M. Matarić, and G. S. Sukhatme. Multi-robot task allocation through vacancy chain scheduling. *Journal of Robotics and Autonomous Systems*, 57(6):674–687, 2009.

[7] V. D. Dang and N. R. Jenings. Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 564–571, 2004.

[8] J. P. Desai. A graph theoretic approach for modeling mobile robot team formation. *Journal of Robotic Systems*, 19(11):511–525, 2001.

[9] M. B. Dias and A. Stentz. A free market architecture for distributed control of a multirobot system. In *Proceedings of the Sixth Conference on Intelligent Autonomous Systems (IAS-6)*, pages 115–122, 2000.

[10] M. B. Dias, R. Zlot, M. Zinck, J. P. Gonzalez, and A. Stentz. A versatile implementation of the traderbots approach for multirobot coordination. In *Proceedings of the Eighth Conference on Intelligent Autonomous Systems (IAS-8)*, 2004.

[11] M. B. Dias, R. M. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: a survey and analysis. *Proceedings of the IEEE*, 94(7):1257 – 1270, July 2006.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[13] B. P. Gerkey and M. J. Matarić. Murdoch: publish/subscribe task allocation for heterogeneous agents. In *Proceedings of the fourth international conference on Autonomous agents (AGENTS-00)*, pages 203–204, 2000.

[14] B. P. Gerkey and M. J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23:939–954, 2004.

[15] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project - tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference on Advanced Robotics*, pages 317–323, Coimbra, Portugal, Jul 2003.

[16] G. A. Kaminka, R. Schechter-Glick, and V. Sadov. Using sensor morphology for multirobot formations. *IEEE Transactions on Robotics*, 24(2):271–282, 2008.

[17] M. Lemay, F. Michaud, D. Letourneau, and J. M. Valin. Autonomous initialization of robot formation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[18] K. Lerman, C. Jones, A. Galstyan, and M. Matarıc. Analysis of dynamic task allocation in multi-robot systems. *Inernational Journal of Robotics Research*, 25(3):225–241, 2006.

[19] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas. Distributed multi-robot task assignment and formation control. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 128–133, Pasadena, CA, 2008.

[20] L. E. Parker. On the design of behavior-based multi-robot teams. *Advanced Robotics*, 10(6):547–578, 1996.

[21] W. Ren and N. Sorensen. Distributed coordination architecture for multi-robot formation control. *Robotics and Autonomous Systems*, 56(4):324–333, 2008.

[22] P. V. Sander, D. Peleshchuk, and B. J. Grosz. A scalable, distributed algorithm for efficient task allocation. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 1191–1198, 2002.

[23] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.

[24] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, pages 209–238, 1999.

[25] T. Sandholm and V. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1):99–137, 1997.

[26] S. Sen and S. Dutta. Searching for optimal coalition structures. In *Proceedings of the Fourth International Conference on Multiagent Systems*, pages 287 – 292, 2000.

[27] K. C. Sevcik. Characterizations of parallelism in applications and their use in scheduling. In *Proceedings of ACM Conference on Measurement and Modeling of Computation Systems*, pages 171–180, May 1989.

[28] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[29] P. Tosic and G. Agha. Maximal clique based distributed group formation for autonomous agent coalitions. In *Coalitions and Teams Workshop, AAMAS*, 2004.

[30] R. M. Zlot and A. Stentz. Market-based multirobot coordination for complex tasks. *International Journal of Robotics Research, Special Issue on the 4th International Conference on Field and Service Robotics*, 25(1):73–101, 2006.

Large Scale Environment Partitioning in Mobile Robotics Recognition Tasks

Boyan Bonev and Miguel Cazorla

***Abstract*—In this paper we present a scalable machine learning approach to mobile robots visual localization. The applicability of machine learning approaches is constrained by the complexity and size of the problem's domain. Thus, dividing the problem becomes necessary and two essential questions arise: which partition set is optimal for the problem and how to integrate the separate results into a single solution. The novelty of this work is the use of Information Theory for partitioning high-dimensional data. In the presented experiments the domain of the problem is a large sequence of omnidirectional images, each one of them providing a high number of features. A robot which follows the same trajectory has to answer which is the most similar image from the sequence. The sequence is divided so that each partition is suitable for building a simple classifier. The partitions are established on the basis of the information divergence peaks among the images. Measuring the divergence has usually been considered unfeasible in high-dimensional data spaces. We overcome this problem by estimating the Jensen-Rényi divergence with an entropy approximation based on entropic spanning graphs. Finally, the responses of the different classifiers provide a multimodal hypothesis for each incoming image. As the robot is moving, a particle filter is used for attaining the convergence to a unimodal hypothesis.**

***Index Terms*—Visual localization, entropy, Jensen-Rényi divergence, classifier, particle filter.**

I. Introduction

MOBILE robotics is a field with increasing number of applications and the range of possible environments for a robot is becoming wider. The capability of learning without the need of human aid is becoming a basic aspect in mobile robotics. Among the variety of machine learning applications to mobile robotics, an essential one is localization. The localization problem consists of estimating the position of the robot in a given map, based on the information provided by the sensors of the robot, such as cameras and range sensors, as well as the odometry of the robot, if available. This work presents a novel approach to image-similarity-based localization. The main purpose of the method is the scalability to large and complex environments, and this aspect is dealt with by dividing the domain in as many partitions as needed.

Cameras provide rich information and its correct interpretation is a complex and open problem. There are two well differentiated approaches to visual recognition: the structural-description models and the image-based models. The first one is usually more complex and task-specific. The image-based recognition, also known as appearance-based recognition, relies on general features extracted from the image, ignoring the structures. In robotics this approach has already been used for localization. For instance, in [1] Menegatti et al. weight omnidirectional samples according to image similarity, to implement a Monte-Carlo localization method for constrained indoor environments. This technique is used for managing multi-modal probability density, for example when the current image matches more than one reference image. Still the complexity and size of the environment restrict the applicability of this method, because the image matching relies on a single similarity function of the Fourier coefficients of the reference images. Any classifier has a limited capacity which limits its scalability to deal with more complex patterns [2].

In this work we propose an unsupervised division of the sequence of reference images in several subsequences of images. For each one of the partitions we associate a similarity measure which yields an appropriate image retrieval result. We take as a similarity measure the euclidean distance in a suitable feature space which is automatically selected from a general set of low-level features. Two major questions arise from this approach: which division to perform on the sequence and, given a test image, which one of the several similarity measures to consider. For dealing with the first problem we make use of Information Theory. We estimate the local Jensen-Rényi divergence [3],[4] among the previous and the next images of the sequence and the divergence peaks determine the limits between two consecutive partitions. The Jensen-Rényi divergence is estimated in the feature space of the images, where the features are a set of low-level filters, similarly to a previous work [5], [6]. The second question we have to deal with, is how to put together the results generated by the different similarity measures. This problem is tackled with a particle filter [7], provided that the robot moves in some direction of the trajectory. In this experimental setup we are assuming that the robot will follow the same trajectory in which the reference images are taken. This assumption is inspired by a previous work [8] in which a robot performs vision-based navigation along corridor-like environments.

The rest of the paper is organized as follows. In Section II we explain the setup of the experiments. In Section III we present our information theoretic approach to data partitioning. Next, in Section IV we explain how we obtain a similarity measure for each different partition. Then in Section V we present a way put the results together and we show some results. In Section VI we conclude presenting our conclusions and future work.

Boyan Bonev and Miguel Cazorla are with the University of Alicante.
Dept. de Ciencia de la Computación e Inteligencia Artificial,
Universidad de Alicante
Apdo. 99, E-03080 Alicante, Spain
E-mail: boyan@dccia.ua.es

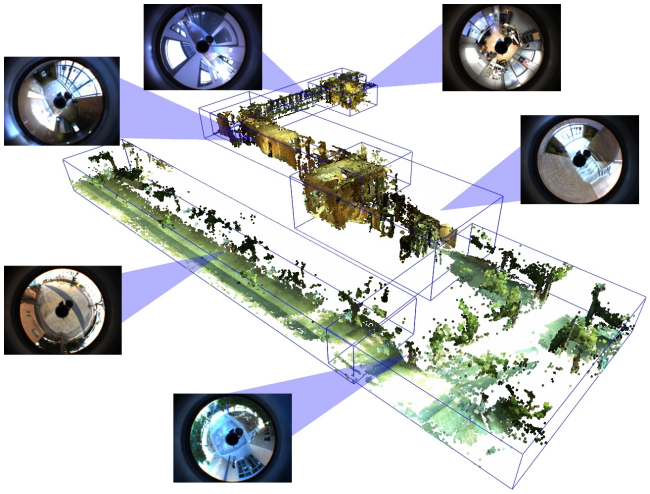

Fig. 1. Representation of the trajectory followed by the robot for taking the reference images. Sample omnidirectional images are shown. The route consists of a laboratory, a narrow corridor, a wide corridor, stairs, a hall, two short outdoor segments and a large one. The 3D-representation is courtesy of Juan Manuel Saez (University of Alicante).

Fig. 2. The omnidirectional mirror is oriented to capture the ground and 360º of the surroundings. This orientation allows rotation invariance when rotating along the vertical axis. The omnidirectional mirror is the Remote Reality's OneShot360 and the video capturing device (mounted on the base of the lens) is a GreyPoint Flea2 firewire camera.

II. Experimental setup

The experimental procedure for the presented system consists of making a camera-equipped mobile robot or a person follow some definite path or trajectory, in order to save video or a dense sequence of images. We call this sequence *reference images* because localization is with respect to them. In the presented experiments the size of sequence is 440 images taken along a $180m$-long indoor/outdoor route which is shown in Fig. 1. In order to navigate, we use the method proposed in [8] because it is also based in omnidirectional images.

Once the images are collected, the system unsupervisedly performs a partition of the sequence. Then, a feature selection process is performed for each partition, in order to optimize each similarity function for its associated interval of images. The bank of filters used for selection consists of rotation invariant low-level filters such as edge detectors and color filters. Rotation invariance is possible because the camera we use is omnidirectional and is vertically oriented, as shown in Fig. 2.

The initial set of filters $\mathcal{F}$ consists of the responses of each image to the following filters:

- Nitzberg
- Canny
- Horizontal Gradient
- Vertical Gradient
- Gradient Magnitude
- 12 Color Filters $H_i,\ 1 \leq i \leq 12$

The filters are applied to 6 different scales of the image. Moreover, the omnidirectional image is divided in 4 concentric rings and filters are also separately applied to each individual ring. This ring division and the whole feature extraction process are explained in more detail in [5] and are illustrated in Fig. 3. Therefore the filters bank $\mathcal{F}$ consists of $|\mathcal{F}| = 17 \times 6 \times 5 = 510$ features. Each image represents a point in this feature space which is invariant to rotation and is much less sensitive to small image variances than the raw

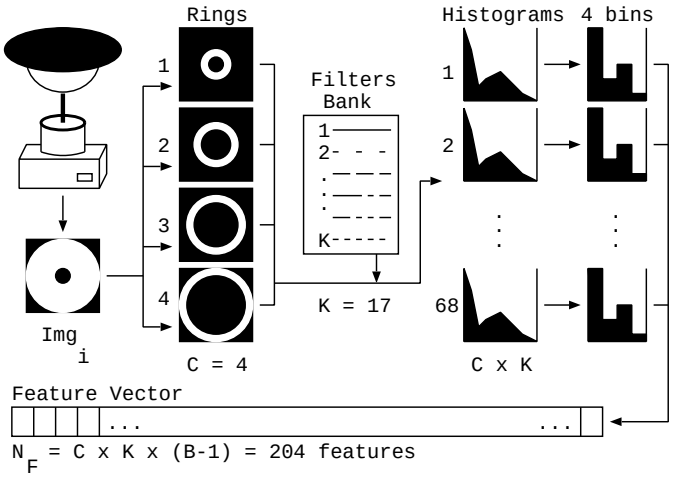

Fig. 3. The feature extraction process.

pixels space. The comparisons between images are performed in this feature space or a subset of it, as detailed in Section IV.

After the learning phase, we perform tests which consist of starting from some random position in the reference trajectory and moving in some direction, which can be forwards or backwards with respect to the direction of the reference trajectory. Localization along a trajectory makes sense in environments where the robot follows only forward/backward paths, like corridors, streets or avenues. Some navigation methods have already been developed for such situations. Concretely in [8] is described a previous work in which corridor-following navigation is performed using the same vision sensor that we use in this work. As the robot is navigating, it takes test

images at a fixed time interval. The first test image taken causes a multimodal response. The next images will make the system converge to a unique hypothesis for the position of the robot, with an error of ± 2 reference images. In the presented experiments about 10–15 images are necessary for achieving convergence. For larger sequences convergence to a unimodal hypothesis would be slower, keeping a low error.

III. Data Partitioning

A. Motivation

Dividing the data in several partitions is a key to scalability. In this work we have to establish a similarity measure capable of indicating the most similar reference image to a new input image. This measure is formulated in terms of a distance between images in an appropriate feature space, further explained in Section IV. This formulation is equivalent to a *K-nearest neighbour* (K-NN) classifier where each reference pattern is a separate class. In the Machine Learning field it is well known that the capacity of a classifier is in a tradeoff with its generalization properties [9]. In other words, a large amount of data requires a high capacity, which incurs on an inaccurate classification of *test* patterns (those which do not belong to the training set or reference patterns). To avoid a classification accuracy decrease with the increase of the amount of training data we divide the data in as many partitions as needed. On the other hand a formulation with a single classifier is unable to handle the *perceptual aliasing* problem, which refers to different states producing similar sensor response. For example, a long trajectory could include corridors with identical appearance. In these circumstances the only way to know the correct location is to have into account the previous images, which is explained in detail in Section V. In the following Subsection we explain the criterion we propose for data partitioning.

B. Partitioning

Finding the optimal partitioning of the sequence of images would involve evaluating each possible subsequences configuration. As the optimal number of subsequences is also unknown, there are

$$\sum_{k=0}^{N} \left(\begin{array}{c} N \\ k \end{array} \right)$$

possible partition sets configurations for an amount of N reference images. In the presented experiments, $N = 440$ which means that there are $2,8392 \cdot 10^{132}$ combinations. The complexity of the problem makes it necessary to use some heuristics for finding a good partition set instead of the optimal one.

The heuristic we propose to use for this problem is based on Information Theory. The idea is to find discontinuities in the sequence of reference images in the feature space $\mathcal{F}$. The reason for this is that each partition has associated its own similarity measure, based on a subset of those features. For example, when the trajectory leaves the laboratory and the corridor begins, there is a significant discontinuity which we are interested in. Information Theory offers tools for finding such interesting places in terms of entropy analysis. Entropy analysis allows us to measure changes in the amount of information in the data, regardless of data's nature. An example can be seen in Fig. 4. In this example a single feature is analyzed in order to show the idea of entropy divergence. However, we do not analyze single features but we estimate divergence in the whole feature space, as explained in the following subsection.

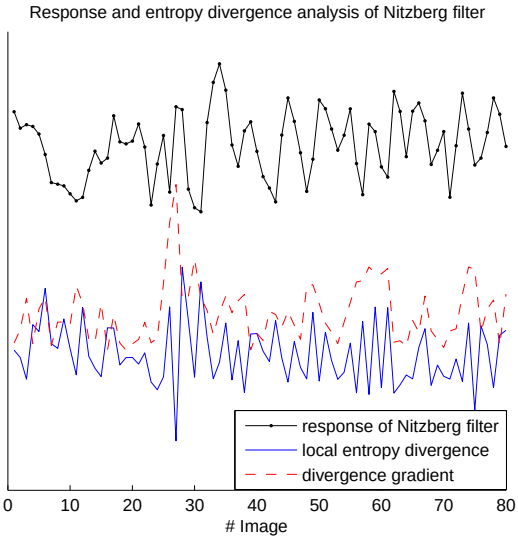

Fig. 4. Entropy divergence analysis of a single feature: the Nitzberg filter for the reference images 1–80. It can be observed that the filter response does not present any significan maxima or minima. However there is a change in the variability near to image #30. The entropy divergence measure lets us notice the change by presenting a high gradient at that point.

C. Jensen-Rényi divergence

In order to calculate the entropy divergence over the whole feature space of the images we need a way of estimating entropy in high-dimensional spaces. Traditionally this has been a drawback and entropy has usually been estimated in one or two dimensions because of the high computational complexity and estimation errors in spaces with more dimensions. However there are methods for entropy estimation which do not depend on the number of dimensions of the data. A widely used one is the estimation of Rényi entropy with *entropic spanning graphs*, described by Hero and Michel in [10]. In [6] we explain the estimation of Rényi entropy and further approximation to the Shannon entropy for calculating the Mutual Information with a feature selection purpose. Also, in [11] *entropic spanning graphs* are used for Mutual Information estimation for image registration. In this work we use the Jensen-Rényi divergence which is a more general criterion than Mutual Information, as shown in [3].

In [4] Hamza and Krim explain the properties of the divergence and show an example of edge detection for image segmentation. They show that it is nonnegative for $\alpha \in (0, 1)$ and it is symmetric and vanishes if and only if the probability distributions $p_1, p_2, \cdots, p_n$ which it measures, are equal, for all $\alpha > 0$. The Jensen-Rényi divergence is symmetric and

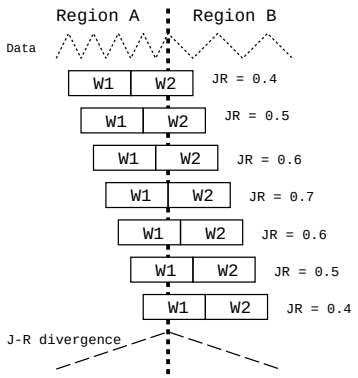

Fig. 5. Sliding window for finding edges with the Jensen-Rényi divergence.

generalizable to any arbitrary number of probability distributions, which is not possible with other divergences, like the Kullback-Leibler one, for example. Also the Jensen-Rényi divergence gives the possibility to assign weights to the distributions.

Although Jensen-Rényi divergence is capable of measuring the divergence among several sets of data, in this work we use it for measuring the divergence between two distributions, for edge detection. We take a sliding-window approach, with two subwindows of equal size. The divergence of the two subwindows W_1 and W_2 is evaluated as the window slides, as represented in Fig. 5. The general formula of the divergence for n probability distributions $p_1, p_2, \cdots, p_n$ is defined as:

$$JR_\alpha^{\vec{\omega}}(p_1, \cdots, p_n) = H_\alpha\left(\sum_{i=1}^{n} \omega_i p_i\right) - \sum_{i=1}^{n} \omega_i H_\alpha(p_i), \quad (1)$$

where $H_\alpha(p)$ is the Rényi entropy, α is the Rényi entropy's parameter, and $\vec{\omega} = (\omega_1, \omega_2, \cdots, \omega_3)$ is a weight vector which satisfies $\sum_{i=1}^{n} \omega_i = 1$ and $\omega_i \geq 0$ and α.

For the case of two subwindows, we have that W_1 slides gradually from region A to region B (Fig. 5). If p_a is the distribution of region A, p_b the one of region B, p_1 the distribution of W_1 and p_2 the distribution of W_2, then we have that the parts of W_2 located in the regions A and B have partial histograms p_a and p_b with weights proportional to the sizes of the intersecting subregions. Let us say that $\lambda \in [0, 1]$ is the fraction of W_2 included in the region B, then $p_2 = (1-\lambda)p_a + \lambda p_b$. This probability distribution is variable depending on the position of the window W. The subwindow W_1 lies entirely in the region A, so its distribution $p_1 = p_a$ and $p = (1-\lambda/2)p_a + (\lambda/2)p_b$. Thus, the Jensen-Rényi divergence in function of λ is:

$$\begin{aligned} JR_\alpha(\lambda) &= H_\alpha\left(\frac{1-\lambda}{2}p_a + \frac{\lambda}{2}p_b\right) && (2) \\ &- \frac{H_\alpha((1-\lambda)p_a + \lambda p_b) + H_\alpha(p_a)}{2} && (3) \end{aligned}$$

For $\lambda = 1/2$ the formula of the window's divergence is:

$$JR_\alpha(p_1, p_2) = H_\alpha\left(\frac{p_1 + p_2}{2}\right) - \frac{H_\alpha(p_1) + H_\alpha(p_2)}{2}, \quad (4)$$

The Rényi entropy (also known as α-entropy) of a probability density function $p(\vec{x})$ is defined as:

$$H_\alpha(p) = \frac{1}{1-\alpha} \ln \int p^\alpha(\vec{x}) d\vec{x}, \ \alpha \in [0, 1[\quad (5)$$

and it can be estimated by building the *minimal spanning tree* $\text{MST}(\{\vec{x}_i\})$ of the data $\vec{x}$ and computing the weighted length of its edges $\vec{e}$:

$$L_\gamma(\{\vec{x}_i\}) = \sum_{e_{ij} \in \text{MST}(\{\vec{x}_i\})} | \, e_{ij} \, |^\gamma, \ \gamma \in [0, D] \quad (6)$$

where d is the number of dimensions of the data. The following Rényi entropy estimator is asymptotically stable and consistent for $D \geq 2$, as showed in [12]:

$$H_\alpha(\{\vec{x}_i\}) = \frac{D}{\gamma}\left(\ln \frac{L_\gamma(\{\vec{x}_i\})}{N^\alpha} - \ln \beta_{L_\gamma, d}\right), \quad (7)$$

where $\gamma = D(1-\alpha)$, N and D are the number of samples and the number of dimensions of the data $\vec{x}$, and $\beta_{L_\gamma,d}$ is a constant not depending on the probability function but on the graph minimization criterion. An approximation [13] that can be used for large d is $\beta_{L_\gamma,d} \approx \frac{\gamma}{2} \ln \frac{d}{2\pi e}$. This approach to Rényi entropy estimation has been successfully used in a previous work [6] where it is explained in more detail.

Finally, even though the α parameter is fundamental in α-entropy, it has a less significant effect on the entropy divergence (Eq. 4). If the misalignment between the distributions could be modeled accurately, then $\alpha = 0$ would correspond to the best choice as it generates a Dirac function at the matching point. In practice this is not a robust selection as a less peaked function is necessary for finding a maximum. On the other hand, $\alpha = 1$ is the most robust choice but it yields the function with least sharp peak. In the case of using the Jensen-Rényi divergence for segmentation purposes, several values of the α parameter have similar performance. In our experiments we set $\alpha = 0.8$.

To sum up, we can perform entropy divergence analysis like in Fig. 4 using a sliding window and calculating the divergence with Eq. 4. The approximation in Eq. 7 makes it possible to work with a large number of dimensions, which in our case is $d = |\mathcal{F}| = 510$ features. A question that arises now is: which size of the window is the most appropriate? It depends on the environment and on the distance between the images. However a multiscale analysis shows up that the discontinuities of interest remain with several window sizes, while those which do not interest us get displaced or disappear. See Fig. 6 where the divergence at 30 window sizes is represented. It is easily observed that at the images 241, 254 and 273, there are gradients in the divergence at all the scales while the other gradients get diagonally displaced. Therefore the criterion that we establish for dividing the sequence in partitions is the presence of strong peaks in the gradient function, for those gradients which are present at different scales. The result of partitioning the whole sequence or 440 images is shown in Fig. 7 where the multiscale divergence gradient is represented and the most peaked points are selected. There are 19 important discontinuities so 20 partitions are established. In Fig. 8 we show the physical position of the

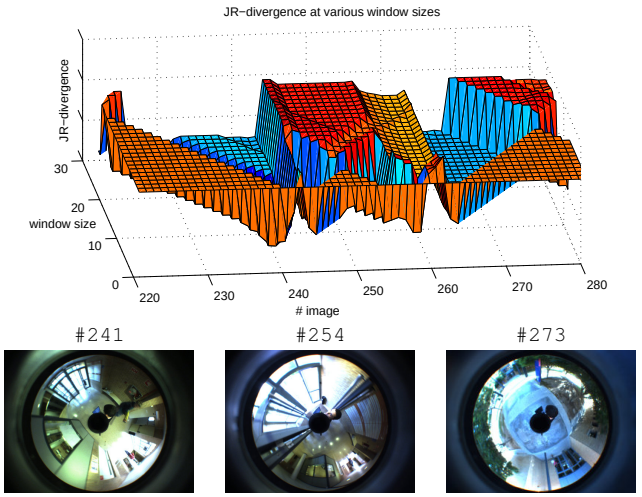

Fig. 6. Top: A representation of the Jensen-Rényi divergence at 30 different scales for some images from the sequence. We can see that the X positions of some discontinuities get displaced at different scales (the scales refer to the window size) while some others persist. We are interested in the latter ones. Bottom: three sample images corresponding to the persistent discontinuities of the upper plot.

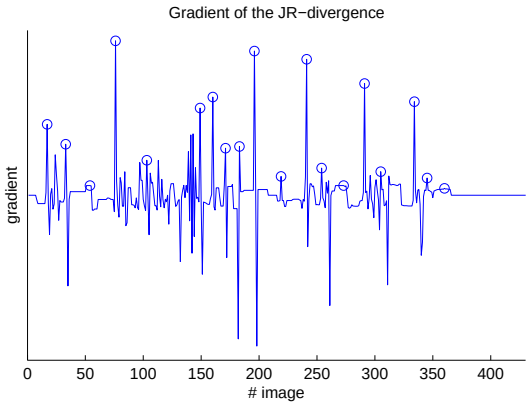

Fig. 7. The gradient of the divergence along the whole sequence of reference images. Some of the peaks are selected as significant discontinuities according to their sharpness.

discontinuities along the trajectory. In the next Section we explain how we perform localization for each single partition.

IV. Localization in each Partition

This Section explains how we obtain a similarity function which is adequate for a definite subsequence of reference images and how we use it for new images. Given a sequence of N images $\mathcal{I} = (I_1, \cdots, I_N)$ which are associated to the linear positions along the trajectory $\mathcal{S} = (s_1, \cdots, s_N)$, and given a test image I_T associated to a s_T position, the objective is to find an image similarity measure $M(I_i, I_T)$ which minimizes the error between the real position of the test image and the estimated position:

$$|s_{\hat{\imath}} - s_T|, \ \forall s_T / s_1 \leq s_T \leq s_N \tag{8}$$

where $\hat{\imath}$ is the index of the reference image $I_{\hat{\imath}}$ which is the most similar (has the shortest distance) to I_T:

$$\hat{\imath} = \arg\min_i M(I_i, I_T) \tag{9}$$

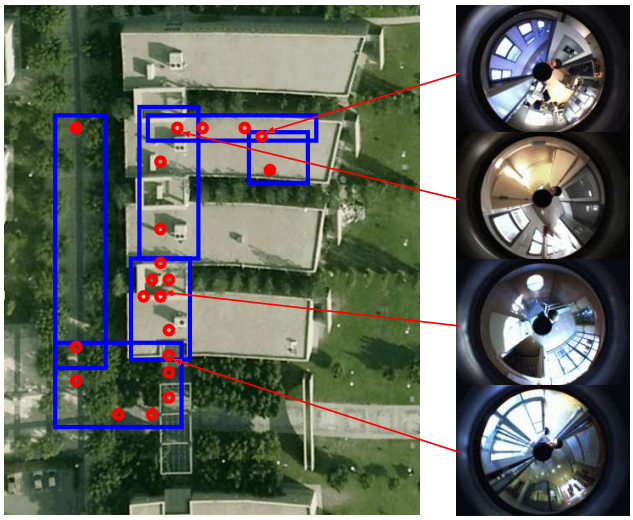

Fig. 8. Left: The reference images which are selected as discontinuities are marked with a red circle. Blue boxes are an analogy with Fig. 1. Right: Some sample images. The third one corresponds to stairs, which cause several discontinuities near to each other.

Provided that the images are in a D-dimensional feature space $\mathcal{F} = \{F_1, \cdots, F_D\}$ (already explained in Section II) we define the dissimilarity measure as the weighted euclidean distance in $\mathcal{F}$:

$$M_{\vec{\omega}}(I_i, I_j) = \sum_{d=1}^{D} \left(\omega_d \left[F_d(I_i) - F_d(I_j)\right]\right)^2 \tag{10}$$

where the weights $\vec{\omega} = (\omega_1, \cdots, \omega_D)$, $\omega_i \in \{0, 1\}$ determine which features are considered and which are not. These weights have to be set for minimizing the objective defined in Eq. 8. Intuitively, this minimization modifies the feature space so that the order of the images in the new space becomes more adequate in the sense that each image is similar to its actual neighbors and it is dissimilar to the rest of the images, as illustrated in Fig. 9.

In order to achieve a good generalization for new images, the maximum number of possible test images have to be considered. According to the definition of the problem (Section II) we have at our disposal only one sequence of N images for the training process. Therefore we have to separate it in train set and test set. The Leave One Out Cross Validation (LOOCV) evaluation method maximizes the number of tests. With this procedure the sequence is divided in a train set of $N-1$ images and a test of 1 image; the evaluation is repeated N times until every image in the sequence has played the test role. The following algorithm iteratively selects important features in a greedy order:

`Input:` $\mathcal{I}, \mathcal{F}, N, D$
$j = 0$
$\omega_i = 0, \ \forall \omega_i \in \vec{\omega}$
`while` $\exists \omega_i \ / \omega_i = 0$
  $\forall i \ / \omega_i = 0$
    $\vec{\omega}' = \vec{\omega}$
    $\omega'_i = 1$

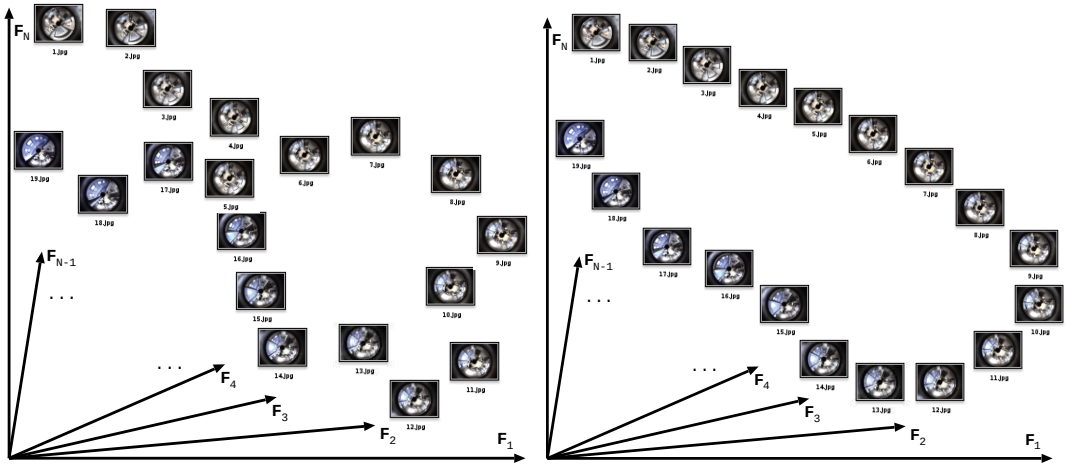

Fig. 9. The feature space is modified so that each image lies near to their actual neighbours, and it lies farther from the rest of the images.

$$\varepsilon_i = \frac{1}{N}\sum_{T=1}^{N} |s_T - s_{\arg\min_k M_{\vec{\omega}'}(I_T, I_k)}|$$
$$\omega_{\arg\min_i \varepsilon_i} = 1$$
$$j = j + 1$$
$$E_j = \min \varepsilon_i$$
$$\Omega_j = \vec{\omega}$$

end

Return: $\Omega_{\arg\min_j E_j}$

The algorithm does not have a stopping criterion. Instead, it keeps on selecting features until all of them (D) are selected, and stores them in order, together with their associated classification errors. Finally, it returns the weights vector which has the lowest associated error.

Once performed the feature selection process, the dissimilarity measure for the set of images $\mathcal{I}$ is applied accordingly to Eq. 10 and the weights $\vec{\omega}$. When a new image arrives, it is said to be closest to some reference image $T_{\hat{t}}$, where $\hat{t}$ is the number of reference image which has the closest distance $M_{\vec{\omega}}$ to the new image, as expressed in Eq. 9. In Fig. 11 are represented the estimations of the 20 different similarity measures, for 220 test images which have not been used for the training process. It can be observed that for a single test image, each similarity function has a different response because each one is trained for a different subsequence of images. See Fig. 10 where a single classifier is trained for the whole sequence of images and for a small range of images (for a single partition). In the following Subsection we explain how we put these results together to obtain a single estimation.

V. Localization in the whole Domain

In mobile robotics localization a very important source of information is the history of previous perceptions of the robot, as well as odometry, if available. When a new perception produces multiple hypothesis, as shown in Fig. 11, history can help to disambiguate and converge to a unimodal hypothesis. We tackle the problem with the classical Monte Carlo Localization (MCL), also known as *particle filter*. This is a Bayesian approach which aims to estimate recursively the

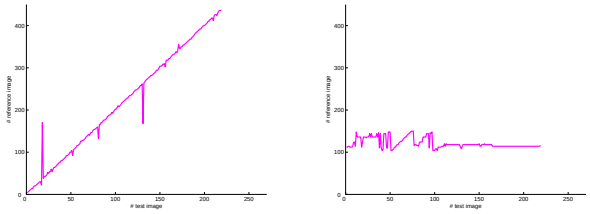

Fig. 10. Left: a classifier trained for the whole sequence of images. Right: a classifier trained for the images in the range 104–148 (a single partition of the whole sequence).

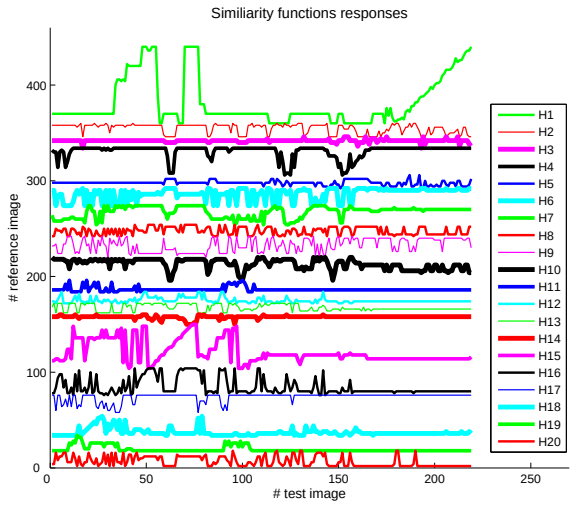

Fig. 11. Responses of 20 different similarity measures, each one trained for a particular subsequence of images. Note the coherence between test and reference images in the diagonal of the plot. Each test image produces 20 different hypotheses, however only one of them is coherent with the previous test images, if they are taken in a sequential order.

posterior distribution $p(s_{T0}, s_{T1}, \cdots, s_{Tk}|I_{T0}, I_{T1}, \cdots, I_{Tk})$, where $s_{T0}, s_{T1}, \cdots, s_{Tk}$ are a sequence of hidden parameters and $I_{T0}, I_{T1}, \cdots, I_{Tk}$ are the sequence of test images observed. Particularly we are interested in a marginal distribution of the posterior, which is called *filtering distribution* and is denoted as $p(s_{Tk}|I_{T0}, I_{T1}, \cdots, I_{Tk})$. From this distribution we can obtain the posterior mode of the state, which in our case is the estimated position with respect to the reference images. This representation is approximate but it is nonparametric, which makes it possible to represent a wide range of distributions.

The MCL algorithm is described and analyzed in [7]. It samples the posterior distribution with a set of *particles* which correspond to positions in the trajectory. We take as many particles as similarity functions we have, however this number could be dynamically changed, according to the complexity of the distribution. For each newly obtained test image, the algorithm first resamples each particle according to a motion model. In our case the motion model is a bimodal distribution with mean ± 1 image and variance 2 images because we assume that the robot moves and takes test images at the same speed as in the training run, and we assume that it can move both forward and backward in the defined trajectory. We do not use odometry in the experiments we present.

Once changed the values of the particle positions, their weights have to be calculated, according to their likelihood. The likelihood is the probability of the perception, given the particle position. For example see Fig. 11 and lets say that a particle was sampled to the position 400 which is the last interval we have in the sequence, then the likelihood will be determined by the response of the similarity function denoted in the plot as $H1.$, which is $\arg\min_j M_{\vec{\omega}'}(I_{Tk}, I_j)$. It can be seen in the plot that if the perception I_{Tk} was actually coming from the image #200 of the test set, then $M_{\vec{\omega}'}(I_{200}, I_{400}) = 0$, so $j = 400$, which is the same value as the particle position and this would be the maximum likelihood possible. Otherwise, let us say that the perception came from the image #190, then according to the same similarity function, $j = 380$ and the likelihood is lower. Finally, if the perception came from some test image number lower than #170, the response would be rather random, as shown in the plot. In this case, even if we obtain a high likelihood for the particle in this iteration, in the following iterations it would get low because the new perceptions would be incoherent with the motion model.

Once calculated the weights of the particle, the algorithm performs an *importance sampling* of the particles. As many particles as needed (a constant number in our implementation) are sampled according to the likelihood of each one of the previously existing particles. This means that particles with a low likelihood will probably disappear, while locations with a high density of particles with a good likelihood will cause a higher concentration of particles. This can be observed in Fig. 12.

The fact that zones with a high likelihood attract the rest of the particles can mislead the algorithm to converge to some incorrect location. When this happens, after a few observations the likelihood of the particles gets very low. To get over such

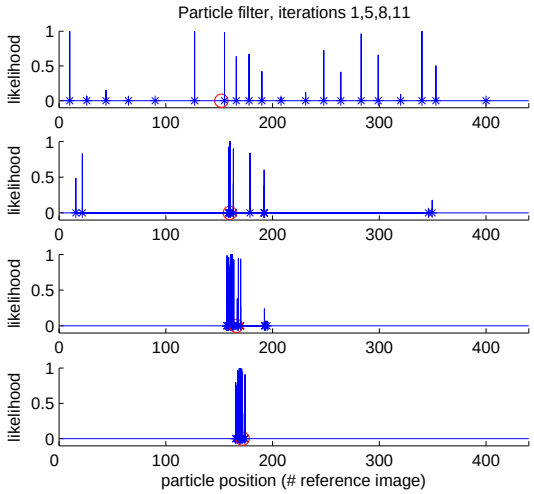

Fig. 12. A Monte Carlo Localization trace of iterations 1, 5, 8 and 11. With the 11-th testing image the convergence to unimodality is achieved. The real position of the test images is represented with a red circle, while the particles are represented with blue asteriscs. The likelihood of each particle is also represented.

situation a simple mechanism is introduced, which takes some particles with a low likelihood and places them randomly along the whole trajectory space. The number of randomized particles depends on the mean likelihood of all the particles. This way if all the particles have a high likelihood, little particles are randomized. This mechanism is also useful for the *kidnapped robot* problem, which consists of taking the robot from its location and make it jump to a different one.

In Fig. 13 is shown the performance of a single similarity function over the whole space of images, in contrast to the use of several functions, Fig. 11. It can be seen that the discontinuities present in a single similarity function are overcome with the partitioning approach.

VI. Conclusion and Future Work

In this work we present a scalable visual localization approach which allows an autonomous robot to get localized over large trajectories. It is an approach based on machine learning and no assumptions are made about the training images. Only one training run over the trajectory is needed for the camera to collect all the reference images. We propose to use an information divergence measure for finding interesting places and partitioning the data. Then we propose to use simple specialized classifiers for each different partition. The off-line learning process for the presented experiment took about 10 minutes both for partitioning and for selecting the features for the different classifiers. Finally a unique localization estimation is obtained with a Monte Carlo sampling method. Classification of each new image can be performed online as its feature extraction process takes several cents of a second and the similarity measures evaluation is very fast. The results are promising, as shows the experiment with an indoor/outdoor trajectory of a 180m length, in which a unimodal hypothesis is usually achieved before the first 12 observations.

A future work is to extend this approach to 2D localization and to topological maps construction and localization. One of

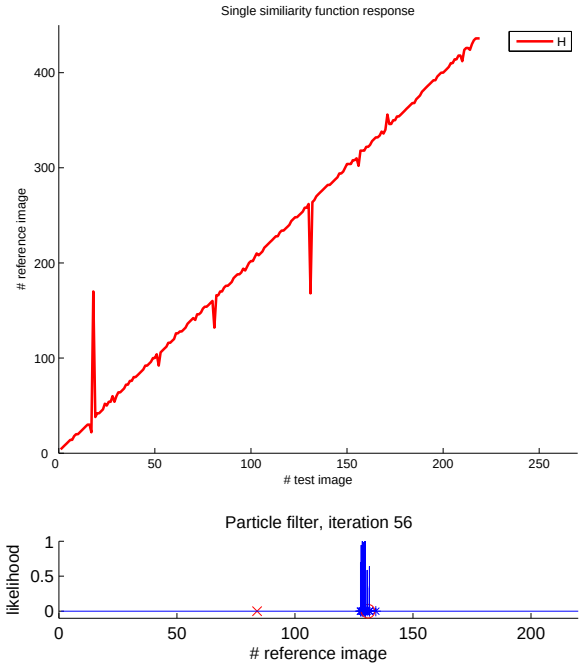

Fig. 13. Top: The response of a single similarity function trained for the whole sequence of images. Bottom: Localization results at the test image #131, given by: a) Single similarity measure for the whole set of images; the response is marked with a red cross. b) Monte Carlo Localization response, the particles are represented with vertical lines. c) Groundtruth, represented with a red circle. It can be observed that for the single similarity measure, the localization result is far from the groundtruth, due to the discontinuity it presents at that point. However the MCL, which uses several similarity measures (Fig. 11), is closer to the groundtruth.

the contributions useful for topological navigation is finding interesting places in a quite data-independent way. These interesting places could be the nodes of a topological map, where a decision about which way to follow could be taken.

ACKNOWLEDGMENT

Thanks to Francisco Escolano and Juan Manuel Saez for their valuable help in the Information Theory field. This research is funded by the project DPI2009-07144 from Ministerio de Ciencia e Innovacion of the Spanish Government.

REFERENCES

[1] E. Menegatti, M. Zoccarato, E. Pagello, H. Ishiguro, *Image-based Monte-Carlo localisation with omnidirectional images*, Robotics and Autonomous Systems, Vol. 48, No. 1, pages 17–30, Elsevier, August 2004.

[2] R.B. Fisher, *An Empirical Model for Saturation and Capacity in Classifier Spaces*, In proceedings of 18th International Conference on Pattern Recognition, Vol. 4, pages 189–193, August 2006.

[3] A. Ben Hamza, H. Krim, *Jensen-Rényi divergence measure: theoretical and computational perspectives*, In proceedings of IEEE International Symposium on Information Theory, page 257, July 2003.

[4] A. Ben Hamza, H. Krim, *Image Registration and Segmentation by Maximizing the Jensen-Rényi Divergence* EMMCVPR 2003 : energy minimization methods in computer vision and pattern recognition. Vol. 2683, pages. 147–163, 2003

[5] B. Bonev , M. Cazorla, *Towards Autonomous Adaptation in Visual Tasks*, In proceedings of VII Workshop of Physical Agents, Las Palmas de Gran Canaria (Spain), pages 59–66, 2006.

[6] B. Bonev, F. Escolano and M. Cazorla *Feature Selection, Mutual Information, and the Classification of High-Dimensional Patterns*, Pattern Analysis and Applications. Pp. 309-319. Vol. 11, No. 3-4. September 2008.

[7] A. Doucet, S. Godsill, C. Andrieu, *On sequential Monte Carlo sampling methods for Bayesian filtering*, Statistics and Computing, Springer Netherlands, Vol. 10, No. 3, July 2000.

[8] B. Bonev , M. Cazorla, F. Escolano, *Robot Navigation Behaviors based on Omnidirectional Vision and Information Theory*, Journal of Physical Agents, Vol. 1, No. 1, pages 27–35, September 2007.

[9] I. Guyon, V. Vapnik, B. Boser, L. Bottou, S.A. Solla, *Capacity control in linear classifiers for pattern recognition* In proceedings of 11th International Conference on Pattern Recognition, volume II, pages 385–388, 1992

[10] A.O. Hero, Bing Ma III, O.J.J. Michel, J. Gorman, *Applications of entropic spanning graphs*, IEEE Signal Process Magazine, Vol. 19, No. 5, pages 85–95, 2002

[11] H. Neemuchwala, A. Hero, S. Zabuawala, P. Carson, *Image registration methods in high-dimensional space*, International Journal of Imaging Systems and Technology, Vol. 16, No. 5, pages 130–145, 2006

[12] A.O. Hero, O. Michel: *Asymptotic theory of greedy approximations to minimal k-point random graphs*, IEEE Trans. on Infor. Theory, Vol.45, No.6, pages 1921-1939, 1999

[13] D.J. Bertsimas, G. Van Ryzin: *An asymptotic determination of the minimum spanning tree and minimum matching constants in geometrical probability*, Operations Research Letters, Vol.9, No.1, pages 223-231, 1990

Design an evaluation of RoboCup humanoid goalie

Juan F. García, Francisco J. Rodríguez, Camino Fernández, and Vicente Matellán

***Abstract*—In this article we describe the ethological inspired architecture we have developed and how it has been used to implement a humanoid goalkeeper according to the regulations of the two-legged Standard Platform League of the RoboCup Federation. We present relevant concepts borrowed from ethology that we have successfully used for generating autonomous behaviours in mobile robotics, such as the use of ethograms in robotic pets or the ideas of schemata, or the use of fixed actions patterns to implement reactivity. Then we discuss the implementation of this architecture on the Nao biped robot. Finally, we propose a method for its evaluation and validation and analyse the results obtained during RoboCup real competition, which allowed us to test first hand how it worked in a real environment.**

***Index Terms*—reactive, vision, humanoid, schema**

I. Robotics control architectures in literature

GENERATING autonomous behaviours in mobile robotics is really a complex problem. In this section we present a non in-depth outline about those robotics control architectures close to our research. We are going to be neither exhaustive, mainly because it would be impossible to describe all control architectures in just one section, nor hierarchycal, since there would be too many criteria to take into account.

Summarising the complex history of the AI, we can state that two main schools of thought have coexisted, the subsymbolic one, interested on modeling intelligence in a level similar to neurons; and the symbolic AI, which models knowledge and planning in data structures that make sense to the programmers that build them. Another way of explaining the difference between both schools is referring to their foundations: Biology in the subsymbolic AI, and cognitive psychology in the symbolic AI [9]. Hybrid systems are a pragmatic approach, where ethology based systems can be included because they successfully integrate deliberative and reactive perspectives in natural autonomous systems.

Hybrid architectures intend to combine reactive and deliberative control, and usually consist of three components: a reactive layer, a planner, and a layer that links the other two. Well known examples of this kind of architecture are AuRA[1], which integrates a A^* planner with schema-based controllers [2], and PRS (Procedural Reasoning System) [6] based on least commitment via plan elaboration postponement.

Teleo-Reactive (TR) program formalism proposed by Nilsson [27] falls also under Hybrid control category. Teleoreactivity in dynamic environments implies a short sense - act cycle. Robots are able react rapidly to commonly occurring situations (such as crash avoidance or refuelling) but their behaviours are also influenced by their goals (hence "teleo").

Juan, Francisco, Camino, and Vicente are with Departamento de Ingeniería Mecánica, Informática y Aeroespacial Escuela de Ingenierías Industrial e Informática, Universidad de León, 24071 León Web: http://robotica.unileon.es e-mail:{jfgars, fjrodl, camino.fernandez, vicente.matellan}@unileon.es

Opportunistic architectures are a subset of the hybrid architectures that take its name from Barbara Hayes-Roth approach to hybrid control [7]. The agents architecture on her system was also made up by three components: an event-triggered reactive level, an strategic planner, and a control process in charge of matching triggered actions with the generated plan. A similar architecture is used in O-Plan [8] where the term "agent" is used to name each of the three modules of the system.

Another implementation of these ideas are RAPs (Reactive Action Packages) proposed by Firby [5]. RAPs were designed to allow the reactive execution of symbolic plans. In this way, each RAP defines different alternatives of execution depending on the environment, and an agenda is used to select the next action to execute. Another approach is the TCA (Task Control Architecture) by Simmons [17], which integrates symbolic plans with real-time restrictions as well as reactive behaviours triggered as exceptions.

In the RoboCup domain, Saffiotti [20] presented the *ThinkingCap* architecture. This architecture was based in a fuzzy approach, extended in [23]. The perceptual and global modelling components managed information in a fuzzy way and were used to generate the next actions.

Also in the RoboCup domain, the architecture proposed by Manuela Veloso et al[21] shows an hybrid hierarchical behaviour-based architecture. This architecture was divided in levels. The upper levels set goals that the bottom level had to achieve using information generated by a set of virtual sensors, which were an abstraction of the actual sensors.

Another successful approach in the RoboCup was the one used in the German Team[22] that proposed a four levels architecture: perception, object modelling, behaviour control, and motion control. The execution starts in the upper level perceiving the environment and finishes at low level sending motion commands to actuators. The behaviour level was made up of several basic behaviours implemented as finite state machines. These finite state machine were written in XABSL language [24], that was interpreted at runtime and let change and reload the behaviour during the robot operation.

Many other concepts borrowed from Ethology have been used in robotics. For instance, homoeostasis, proposed as mechanisms for action selection by T. Tyrrell [15]; or the flies balancing optical flow in both eyes to local navigation [4]; gestalt perception, and the use of visual perceptive invariants, as the ones discovered in the cormorant fishing [14], that can make easier the goal of developing robotic behaviours, etc. These works have also been applied to modern humanoids [3].

The foundation of the work presented in this paper is JDE (*Jerarquía Dinámica de Esquemas*) [9], an etho-inspired architecture where behaviour is organised as a dynamic hierarchy of

independent schemata. This architecture is hybrid in nature, so it is closely related to the other hybrid approaches previously enumerated.

Besides theoretically describing the architecture, we have also implemented it in a robot in order to put it to the test in a real environment. The chosen scenario was the RoboCup (Robotic soccer WorldCup), an international research and education initiative, which has put forward a standard problem to promote the research on artificial intelligence and intelligent robotics.

In particular, the work described in this paper has been tested in the Standard Platform League (SPL) during German Open 2009[1] and Robocup 2009[2]. In this league, all teams use the same hardware platform, the Nao robot (see figure 1). These robots are manufactured by Aldebaran Robotics, so the focus of this competition is on the software controlling the robot.

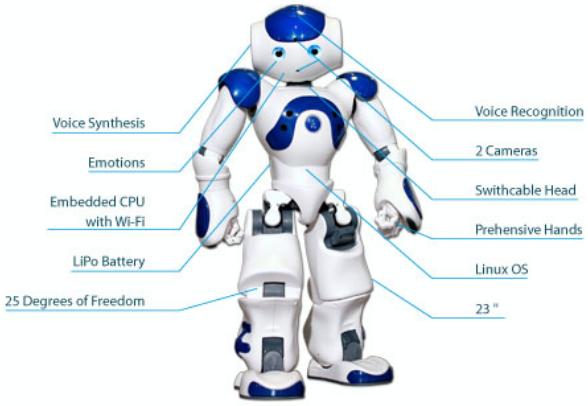

Fig. 1. Nao robot (figure copyrighted by Aldebaran Robotics)

Nao robot is a 21 degrees of freedom humanoid, whose height is 57 cm. and its weight is around 4.5 Kg. It has two 30 fps video cameras located in the forehead and in the mouth, each one with a maximum resolution of 640x480, but they cannot be used simultaneously. The switch between cameras takes too long and the field of view is scarcely overlapped so they are not capable of stereo vision.

Control is managed on-board using a x86 AMD Geode chip at 500 MHz, 256 MB of SDRAM memory and 1 Gb of flash memory that can be upgraded. It also has got WiFi (802.11g) and Ethernet connections. Concerning the sensors, apart from the cameras, it has 2 gyroscopes and 3 accelerometers, 2 bumper sensors in the feet, and 2 ultrasonic sensors in the chest.

The rest of the paper is organised as follows. Second section describes the architecture we propose. In the third section we present a software implementation for our architecture. In the fourth section, we propose a method to analyse and validate our proposal. Finally, in the last section, the results obtained with this architecture and its performance in the RoboCup German Open and in the Robocup 2009 Graz are analysed and also future works are enumerated.

[1]http://www.robocup-german-open.de/en
[2]http://www.robocup2009.org/

II. An Ethological Inspired Architecture

Our architecture is based on ethological principles that exhibit the same features of the hybrid ones previously described, that is, deliberative and reactive capabilities. The two main principles of this architecture are the decomposition of the control problem into behavioural units named components, and the generation of behaviour by building dynamic hierarchies. Both are detailed in next sections.

A. Components

Our approach is based on the assumption that complex behaviour can be obtained by combining simpler "components" inspired by ethological schemata as defined in [10]. These components perform a specific task in an iterative way and at a controlled frequency. They may send commands to actuators, process data from sensors, or activate/deactivate and modulate other components creating a hierarchy.

When activated, a component creates its data and processing structures and starts its iterations. It can keep its state from one iteration to another or change it depending on its functionality and the system stimuli (internal or environmental information). When a component is deactivated, all its descendants (all the components the currently component becoming inactive had activated) must also be finished.

A group of components which perform subroutines of the same task are grouped in so called controllers which functionality is explained in section II-C and their implementation in section III.

B. Dynamic Hierarchy

Components are organised in hierarchy in order to generate more complex behaviours. High level components activate low level components, and all of them run concurrently. The hierarchy is dynamic in the sense that currently active modules will be different depending on the situation.

All schemata in the same level are mutually exclusive, which means only one schema per level can be active at any given time. Also, before activating any component, an ancestor of it in the immediately superior level has to be already active. First restriction helps minimising the risk of trying to perform contradictory tasks, improving system stability: for instance, "move" and "save" schemata are in the same level and both of them send commands to the robot's actuators and servos; if they were activated simultaneously, the result of combining these commands would be unpredictable, and the robot would probably fall. Second restriction guarantees that all prerequisites for the task to be performed are fulfilled before activating the component.

Upon deactivating a component, all its descendants will also become inactive. Each schema or a whole branch of linked schemata can be activated (or deactivated) at any given time to achieve the desired functionality, completely deactivating a previously working set of schemata if necessary.

This makes an improvement to the initial JDE assumption which establishes that every single schema in a certain active hierarchy has to be deactivated one by one before starting a new one.

Components use a common shared memory space to read its inputs and write its outputs. The upper level component connects the output with the inputs of the modules it activates. This way a low level component could be reused by another high level components which could decide to connect the low level components in a different way. All these inputs and outputs define the system information flow, which basically consists of internal (component generated) or external (from the environment) stimuli. All components output are then internal stimuli, while their input can be either an internal or external stimuli depending where it comes from.

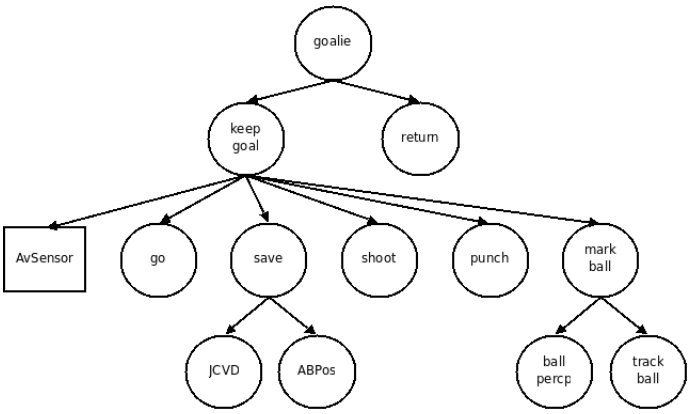

Fig. 2. Goalkeeper modules

Figure 2 shows an example of hierarchy, the goalkeeper behaviour schemata.

C. Controllers

As already explained in section II-A, a controller is a group of schemata which perform subroutines of the same task. The components are grouped to simplify the overall structure of the architecture: it is an effective way to reduce the information flow present in the system.

Information, as we explained in the previous section, consists of external or system internal stimuli which would cause either activation or deactivation of a given component. The main reason to have controllers and not individual ungroupped components is not having to consider an input and output information channel per component of the system at any given time. Instead, information is brought to each controller, and it will then be redirected to the concrete component which is designed to react to it.

Basically, a controller oversees the activation and deactivation of its components redirecting the information flow it receives and produces. Each controller is able to communicate with other controllers coexisting in the system to which it is directly connected the same way isolated components do.

Besides the conceptual simplicity explained, there is no real difference among a controller and a group of components. We describe the controllers we use, its functionality ,and its implementation in section III.

D. Architecture characterisation

The presented architecture shows both deliberative and reactive properties, so it is a hybrid architecture in the classic definition. The set of all possible connections among components and their organisation in different levels, as shown in figure 2, give the system its deliberative nature. In this figure, circles are components, while squares represent system's sensors which provide inputs from the environment (in this case, only the camera sensor is shown). The higher the level, the more abstract and complex behaviours it contains.

The architecture is reactive during the hierarchy activation phase previously explained in II-B: the set of active components will vary depending on the situation, with only those useful for the current behaviour being active. We will give two examples of its reactive nature in next section.

Please note that even if the hierarchy defined by active schemata in a given situation is dynamic (varies depending on the task at hand) - reactive behaviour - the available connections among components and their organisation in levels is fixed and previously established - deliberative architecture -.

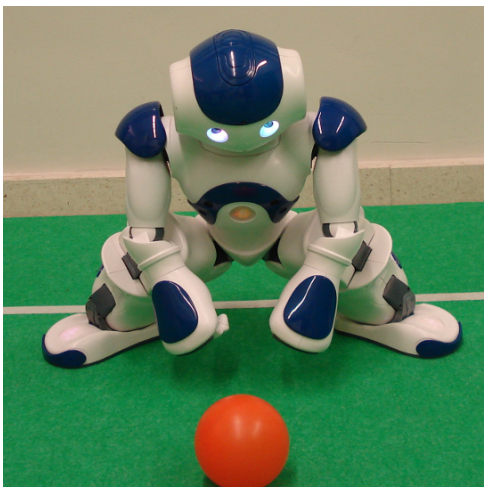

Fig. 3. JCVD defensive movement schema implemented on real Nao robot

III. Implementation

We are interested in testing our architecture in order to prove its functionality. To do so we have chosen to model a goalkeeper behaviour.

A. Components

The components are the ethological schemata which model all the possible actions the goalkeeper needs. We have the following components (with each component's name being pretty much self-explanatory about their functionality):

- *Goalie*: Represents all kind of high level behavioural decisions which a goalkeeper would perform during a match, either specific to its role (eg: punch out the ball), or not (perception tasks).
- *KeepGoal*: Represents all kind of high level behavioural decisions which are specific to a goalkeeper's role.

- *MarkBall*: Tries to keep the ball inside the robots visual field.
- *BallPerception*: Looks for the ball in a given image.
- *TrackBall*: Moves the robots head so that the ball stays in the centre of its visual field.
- *Go*: Makes the robot walk to a given position.
- *Return*: Makes the robot walk to the centre of its keep.
- *Save*: Performs a defensive move to try to stop the ball.
- *JCVD*: A wide-area but slow defensive move intended to prevent a goal.
- *ABPos*: A fast but small-area defensive move intended to prevent a goal.
- *Shoot*: Performs a kick to clear the ball.
- *PunchOut*: Punches out the ball.

B. Controllers and NaoQi Layer

Task related components are grouped into controllers. The controllers we have implemented, its main functionality, and the components they include are:

- `Goalkeeper Controller`: Takes high level decisions about what to do at any given time: look for the ball, move or try to prevent a goal. Includes *Goalie* and *KeepGoal* components.
- `Scanner Controller`: Moves the robot head in order to look for the ball and gets and analyses images from the robot's camera. Includes *MarkBall*, *BallPerception* and *TrackBall* components.
- `Walk Controller`: Allows the robot to walk in different directions. Includes *Go* and *Return* components.
- `Save Controller`: Performs defensive positions intended to stop or clear the ball. Includes *Save*, *JCVD*, *ABPos*, *Shoot* and *PunchOut* components.

Those controllers have been used, as already introduced in section II-C, to reduce the system complexity. By grouping components which take part in the same task we reduce the amount of information channels to be considered at any given time. For instance: *BallPerception* and *TrackBall* components are meant to work with visual information (the first one will look for the ball in any image obtained by the robot camera while the second one will try to centre it in the field of view once it has been found). There is no reason then to use two different information channels carrying the same visual information, so we group them in a controller which we call `Scanner Controller` which will receive this information and then redirect it to the component which actually needs it.

To be able to control the robot, we will use the software it provides, called NaoQi. NaoQi is a proprietary SDK which allows us to access robot sensors and actuators by using the modules it provides. We can not consider that NaoQi modules define a real controller since they implement very different functions, from internal memory management to servo motors control, and thus are not really task-related. However, for simplicity reasons, we will represent all these modules together grouped in what we call `NaoQi Layer`. Also, the only conceptual difference between NaoQi modules and the rest of our components is that components in the NaoQi layer are platform specific, that is, they are part of Nao robot's software. The hierarchical relation between all the controllers can be seen in figure 4.

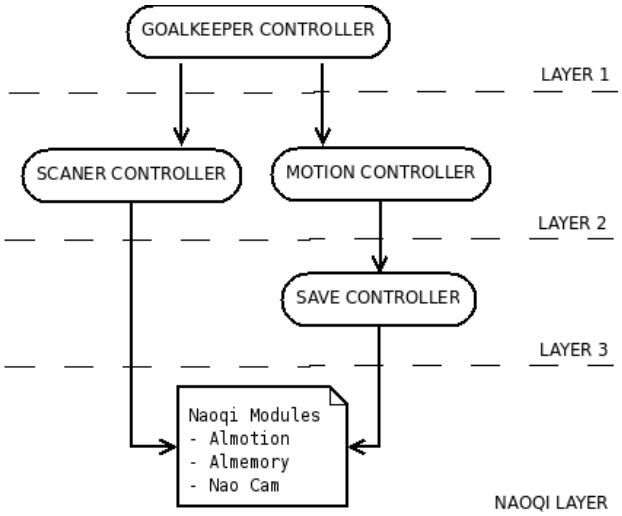

Fig. 4. Goalkeeper controllers

C. Hierarchy

The whole static hierarchy, which consists of all implemented components, can be seen in figure 2, and the controllers which they are part of are shown in figure 4. This static hierarchy represents the deliberative nature of the architecture. The components are organised in levels, with those related to high level tasks occupying the top ones while other more being in the lower levels. The lines connecting components represent the hierarchical relation between them.

The hybrid nature of our architecture can be better illustrated by two examples of generation of autonomous behaviour for our RoboCup goalkeeper.

Example I. Let's assume only scanning and basic saving positions modules are available during a real match (deliberative architecture). Given this situation, the goalkeeper would just activate the *TrackBall* component (after activating the needed ascendant schemata to reach it) - reactive architecture - (see figure 2), thus activating *KeepGoal* and *MarkBall* on its way down to it. It would also eventually try to stop it if it comes too close to the keep by going down the hierarchy activating *KeepGoal*, *Save* and finally *ABPos* (a static defensive position)..

Example II. Let's suppose all schemata are available during the match. In this case, the goalie, once the ball has been found, would perform side steps to position itself in front of the ball, activating *Keep_goal* and *go* schemata. It could even clear the ball activating the *punch-out* component (which is also connected to *Keep_goal*) if the ball comes close enough.

We have two videos[3][4] in our web that show these examples working in a real Nao humanoid. Both videos show the whole tree of components, with the active components displayed in a lighter color. The Nao robot appears to the left of the media

[3]http://robotica.unileon.es/mediawiki/videos/save.swf
[4]http://robotica.unileon.es/mediawiki/videos/movementAndSave.swf

player, performing each action enumerated in both scenarios, while we can observe its internal architecture displayed at the right side. The relation among components and the dynamic hierarchy resulting from their synchronisation are also represented: some schemata are activated when required by the situation while others no longer necessary become inactive.

D. Real world restrictions

When implementing our architecture in a real robot some issues were raised. For instance, not all actions can be instantly stopped (specially those related to movement) to start a new one. This affects not only robots also humans: just imagine you are running and suddenly decide to lay on the floor; you better slow down and stop moving before trying to do so or you will end up rolling on the floor. Applied to our component based design, this means component deactivation will always have a time cost. It is not possible to model this cost because it depends on the current situation.

In most cases, deactivation times are so brief that can be ignored, such is the case of decision-related components like *KeepGoal*, *Save*, *MarkBall* or *TrackBall* (not much time is necessary to decide you are no longer interested in defending your keep or tracking the ball). However, time cost for movement-related ones like *Go*, *ABPos* or *JCVD* (see figure 2) are not negligible, as shown by figure 5. Although these time costs may seem too high, imagine for a moment the time it would take to a human to stop running, fall to the floor to try to stop the ball, and then get up and start running again.

We need to perform some actions before fully deactivating any of the movement-related components so that the robot is not left in an unstable position and thus becomes suitable to fall. We could consider that some sort of cooldown timer is set preventing any new schema activation until the last deactivated schema ensures the robot has reached a stable state. As a result, all schemata have an associated deactivation cost.

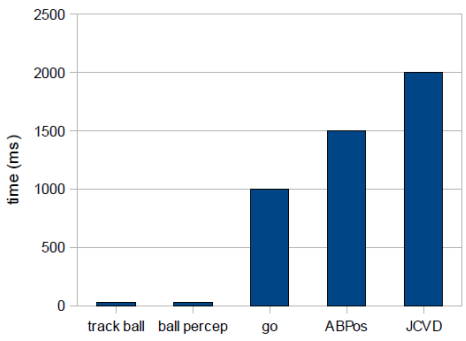

Fig. 5. Deactivation time cost for some schemata

Taking these times into account, some sort of high level deliberative mechanism is necessary for behaviour planning: It is necessary to evaluate the advantages (goal achieved) and drawbacks (in terms of time cost) to deactivate a schema in order to activate a new one. Should I really stop running to comb my hair if I am running to try to catch the bus?, should I stop running to tie my shoes if I am, again, trying not to lose the bus? The answer to first question is obviously "no" since my goal is to get in time before the bus leaves, but it would probably be "yes" to the second one, since it may not be worth taking the risk of falling.

In the robotics world, and specifically in the Robocup environment, we also have plenty of situation which illustrate this kind of situations. For instance, imagine the robot is moving sideways (see figure 6, (1)). When close enough to the ball, the `Goalkeeper Controller` decides it should stop and try to block it by using a fast defensive move (*ABPos* or *JCVD*). The *Go* schema should be deactivated, which would consume a second (see figure 5). As soon as *go* is inactive, *Save* is activated, and then *JCVD* becomes active too (labelled as (2) in figure 6). If the ball suddenly moves away from the goalie (for whatever reason), it will have to stand up and move again. *JCVD* schema would be deactivated, which would consume 2 seconds (it takes some time to get up from the floor). Then, *Save* schema would get inactive (barely instantly since it is just a decision related schema) and finally *go* could be reactivated (see (1) in the same figure). So basically, a "move - stop - save - get up - move" sequence would take more than 3 seconds to be performed in reality (without taking into account the time the save would take per-se), while theoretically those times were neglected.

IV. Architecture analysis and validation

For the Standard Platform League, with the Naos being a relatively new addition, the level of play of many teams is not yet that sophisticated. Our goalkeeper was not called upon to save many goals and so it is difficult to assess its effectiveness relaying just on the results obtained during the championship. It is also difficulty to assess how well would the robot perform without this architecture. Keeping these limitations in mind, we are trying to evaluate the architecture the most objective way possible. To do so, we review all levels presented in the "4+1 View Model of Architecture" by Krutchen [25], using the norm ISO 9126, an international standard for software quality evaluation:

1) **Logic level**. High level programming allowed by schemata and behaviour units usage and low level details being hidden thanks to NaoQi both improve abstraction. The architecture makes it easier to understand already developed behaviours and actions and simplyfies the process of adding new ones, so it complies with "usability" characteristic of ISO 9126. Although it doesn't directly guarantees efficiency, it makes it easier to achieve it since every module can be tested and improved its own.
2) **Processes level**. The goalkeeper behaviour is split in different levels, with every level performing actions independently form the rest (movement, vision, etc.). This makes concurrency management much easier and at the same time it simplyfies the process of adding a new behaviour (for instance, a new kick o new scan mode) or even a whole new controller (for instance, a localization controller) without interfering with the already existing ones. ISO 9126 "Security", "interoperability", and "sta-

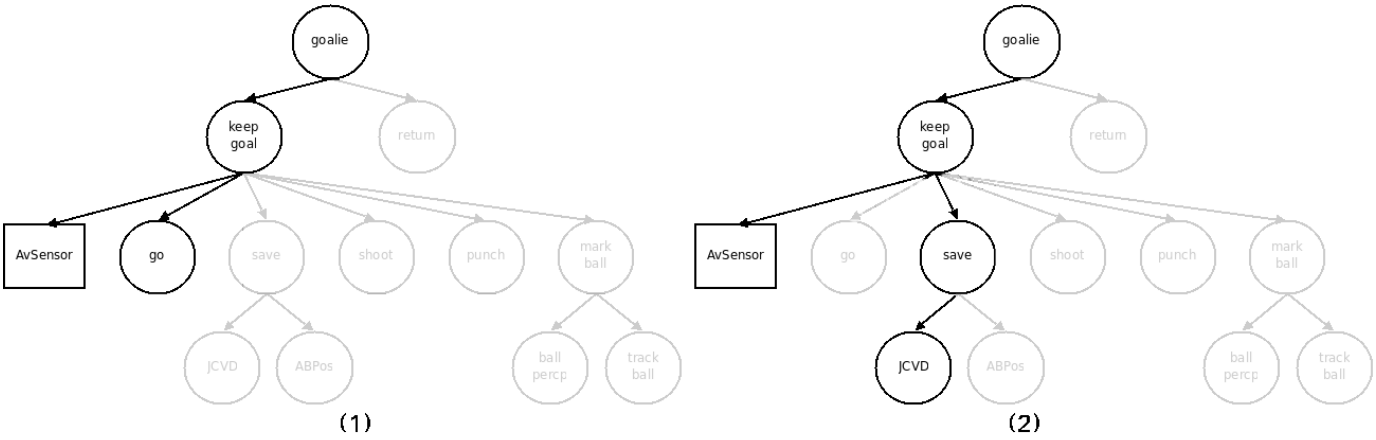

Fig. 6. Schemata active during a "move (1) - save (2) - move (1)" sequence

bility" categories are then maximised when using this architecture.

3) **Development level**. To evaluate development advantages, that is, how this architecture makes the programming of the Nao easier or how behaviours are more quickly developed when using it, we suggest using cocomo (COnstructive COst MOdel), a mathematical empirical model for software costs estimation [26].
4) **Hardware**. The most interesting aspect of our architecture about hardware is that all platform specific calls and functions are contained inside NaoQi layer. Since this layer is developed and maintained by the Nao's company (Aldebaran), hardware optimised usage is taken outside of the architecture and solely relies on their external development. The existence of the NaoQi layer also means that we could use this very same implementation except for that layer for any other robot, which complies with the "potability" category of ISO 9126.
5) **Performance of the four previous levels when working together**. The best way to evaluate performance of architecture as a whole is put it to the test. For that reason, in section IV-A a battery of goalkeeper specific tests is proposed. The matches played during Hannover German Open and Graz Robocup are also a good benchmark themselves.

A. *Goalkeeper behaviour specific tests*

The architecture described has been used to model a goalkeeper behaviour. The best way to test its elements is to check how well a robot with an implementation of it performs the goalkeeper rol. The most relevant tasks fulfilled by a goalkeeper are then put to the test: fast movement over the field, ball perception, ball tracking, proper positioning (deliberative part), and goal saving (reactive part). Since movement speed, and movement in general, is dependant on NaoQi primitives, as it was previously stated in section III-B, it is not taken into account. In order to test the rest of them, the following benchmark is proposed:

Only one side of the field is used. A set of markers are placed on it: they are placed at 4 rows and 5 columns, first row at 50 cm from the keep line and the rest 50 cm from the previous one. The markers are labelled as (m, n), with m (rows) ranging from 0 to 3 and n (columns) from 0 to 4, with marker $(0, 0)$ being the one at the top-left corner when looking at the keep and marker $(0, 2)$ being positioned exactly in front of the robot, at 50 cm from the keep's line. All markers in the same row are also positioned 50 cm from the adjacent ones. Fig. 7 shows a top view of the benchmark proposed.

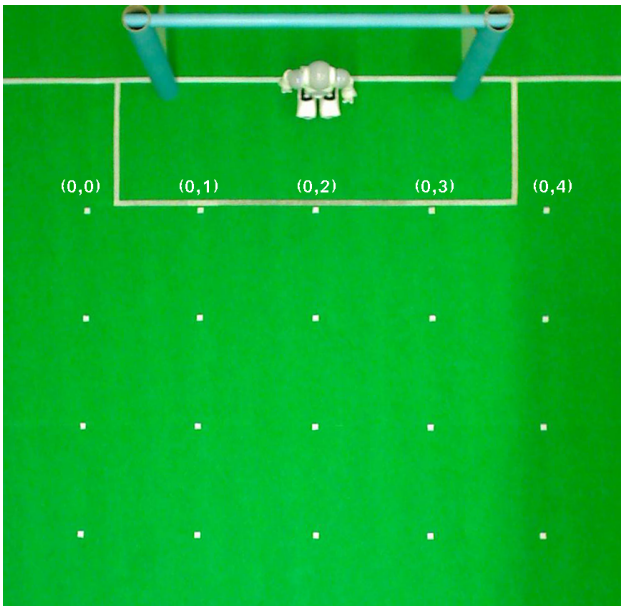

Fig. 7. Markers used for testing

A 130 cm length ramp with variable inclination from 6% to 24% is used to perform the test. The ball can be placed on three different positions on top of it, at 50, 100 and 130 cm from its lower part, which is placed on every marker in the field. For every marker, the angle to the keep can be modified from $-\frac{\pi}{4}$ rad to $\frac{\pi}{4}$ rad with steps of $\frac{\pi}{16}$ rad, which results in nine possible orientation for every marker: $\{-\frac{\pi}{4}, -\frac{5\pi}{16}, -\frac{\pi}{8}, -\frac{\pi}{16}, 0, \frac{\pi}{16}, \frac{\pi}{8}, \frac{5\pi}{16}, \frac{\pi}{4}\}$. This allows us to test nine different orientations and three different ball starting position for every marker; since twenty markers are placed in the field, it means the benchmark

includes a total of 540 different shots. Shot speed can also be modified depending on the ramp's inclination.

To perform the test, the goalkeeper stands in the middle of the keep when started. The ramp is then positioned at every marker (different ball positions on top of it, and different angles and inclination are used for all of them). Once the goalkeeper locates the ball and starts tracking it, the ball is realeased from its initial position and thus approaches the keep. For every shot, the goalkeeper success rate in every category evaluated (ball perception, ball tracking, proper positioning, and goal saving) is measured.

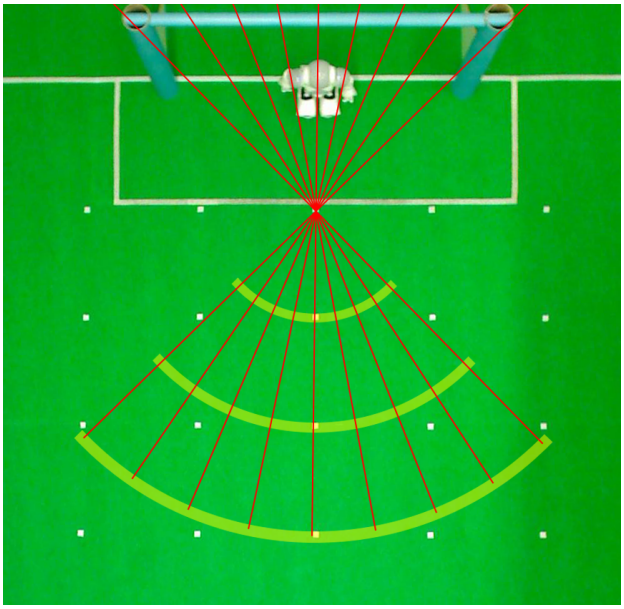

Fig. 8. Possible orientations from marker $(0, 2)$

To evaluate the performance of the goalkeeper, only markers $(0, n)$ from first row were used to test the architecture. Ramp inclination was also fixed at 6%. All nine possible angles and the three starting ball positions on top of the ramp were used. This means 135 different shots were used to test our behaviour. Every shot was repeated ten times to ensure more reliable results, which gives us a total of 1350 measures. Fig. 8 shows the different trajectories possible from marker $(0, 2)$ and Table I shows the success rates (in %) obtained for every characteristic measured and every test performed. Ball perception and tracking is almost flawless, and positioning is also good, specially for central and side markers, that is, $(0, 0)$, $(0, 2)$, and $(0, 4)$. Positioning being not very accurate for markers $(0, 1)$ and $(0, 3)$ (see Fig. 7) is a consequence of far post shots from these positions: the goalkeeper tries to always position itself in front of the ball, which is better to intercept shots aimed at the near post but worse for far post shots, which mostly end up in a goal.

V. Conclusions and Further Work

In this paper we have presented a hierarchical architecture, borrowing concepts such as components (schemata) or dynamic hierarchies from ethology and adding others like the controller concept to simplify the architecture design and the information flow inside the system.

This architecture has been implemented on the software infrastructure offered by the NaoQi development environment for the Nao humanoid.

The implementation has been evaluated using the "4+1 View Model of Architecture" by Krutchen [25]. To further validate its performance in real gameplay environment, a benchmark of goalkeeper specific behaviours has been proposed. The architecture was also tested during RoboCup German Open (April 2009) and RoboCup Official tournament (July 2009). The results obtained in both the benchmark and the competitions show the correctness of the approach:

- The architecture mantains a goalkeeper behaviour for the whole test or match (20 minutes for the former and 10 minutes for the later, both without human intervention).
- The architecture adapts to dynamic match situations, mainly ball position.
- The architecture implements the goalkeeper behaviour using a deliberative structure for planning (self-positioning) and a reactive control mechanism to respond to environmental and internal stimuli (goal saving).
- The architecture allowed the robot to track the ball nearly 100% of the time, position itself properly 84% of the time, and save goals from 62% of the trajectories tested.

Future works envisioned are improving reactive nature of the system via gaze control implementation to react to other elements apart from the ball, and testing architecture's suitability to more complex behaviours that include collaboration with other robots. Minor improvements in goalkeeper positioning behaviour to make it less vulnerable to far post shooting would also be interesting given the results obtained in the tests.

Acknowledgement

We want to express our gratitude and acknowledgement to all members of our team, in particular to the members of the robotics group of the Rey Juan Carlos university for their support, help, and inspiration of the ideas described in this paper.

The authors would like to thank the Spanish Ministry of Innovation for its support to this project under the grant DPI2007-66556-C03-01 (COCOGROM project), and Junta de Castilla y León and European Social Fund for their support to Juan F. Garcia.

References

[1] R. C. Arkin, T. Balch. *AuRA: Principles and Practice in Review*. Journal of Experimental and Theoretical Artificial Intelligence, 9(2-3), pp. 175–188, 1997

[2] R. C. Arkin. *Motor Schema-Based Mobile Robot Navigation*. The International Journal of Robotics Research, 8(4), pp. 92–112, 1989

[3] S. Chernova, R. C. Arkin. *From deliberative to routine behaviours: a cognitively-inspired action selection mechanism for routine behaviour capture*. Adaptive Behaviour Journal, Vol. 15, No. 2, pp. 199–216, 2007.

[4] A. P. Duchon, W. H. Harren, L. P. Kaelbling. *Ecological robotics*. Adaptive Behaviour 6 (3/4), special Issue on Biologically Inspired Models of Spatial Navigation, pp. 473–507, 1998.

[5] R. J. Firby. *Task networks for controlling continuous processes*. In: Proceedings of the 2nd International Conference on AI Planning Systems AIPS-94. Chicago, IL (USA), pp. 49–54, 1994.

marker	ball perception	ball tracking	proper positioning	goal saved
(0,0)	100%	100%	100%	78%
(0,1)	100%	97%	61%	50%
(0,2)	100%	99%	98%	58%
(0,3)	100%	97%	63%	54%
(0,4)	100%	100%	100%	72%

TABLE I
GOALKEEPER'S BEHAVIOUR TEST RESULTS

[6] M. P. Georgef, A. L. Lansky. *Reactive Reasoning and Planning*, Proceedings of AAAI-87 Sixth National Conference on Artificial Intelligence, Seattle, WA (USA), pp. 677–68, 1987.

[7] B. Hayes-Roth. *Opportunistic control of action in intelligent agents*, IEEE Transactions on Systems, Man, and Cybernetics, Vol 23:6, pp. 1575–1586, 1992.

[8] K. Currie, A. Tate. *O-Plan: The Open Planning Architecture*, Artificial Intelligence, Vol. 52:1, pp. 49–86, 1991.

[9] J. M. Cañas, V. Matellán. *From Bio-inspired vs. Psycho-inspired to Etho-inspired robots*, Robotics and Autonomous Systems, Vol.55, Num. 12, pp. 841–850. DOI:10.1016/j.robot.2007.07.010

[10] J. M. Cañas, J. Ruíz-Ayúcar, C. Agüero, F. Martín. *JDE-neoc: component oriented software architecture for robotics*, Journal of Physical Agents, Volume 1, Number 1, pp. 1–6, 2007.

[11] J. F. García, F. J. Rodríguez, C. Fernández, V. Matellán. *Designing a minimal reactive goalie for the RoboCup SPL*. WAF 2009 (Workshop de Agentes Físicos) Cáceres (Spain), 2009.

[12] A. N. Allen, J. C. Shaw, H. A. Simon. *Report on a general problem-solving program*, Proceedings of the International Conference on Information Processing, pp. 256-264, 1959.

[13] K. Lorenz. *Foundations of ethology*, Springer Verlag, New York, 1981.

[14] D. McFarland, T. Bösser. *Intelligent Behaviour in Animals and Robots*, MIT Press, ISBN: 0-262-13293-1, 1993.

[15] T. Tyrrell. *An evaluation of Maes' bottom-up mechanism for behaviour selection*, Journal of Adaptive Behaviour, Vol. 2, Num 4, pp. 307–348, 1994.

[16] G. Walter. *An imitation of life*. Scientific American, 1950.

[17] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, J. O'Sullivan, M. Veloso. *Xavier: Experience with a Layered Robot Architecture,* Agents '97, 1997.

[18] A. Stoytchev, R. C. Arkin. *Combining Deliberation, Reactivity, and Motivation in the Context of a Behaviour-Based Robot Architecture*, In Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation. 290–295. Banff, Alberta, Canada, 2000.

[19] R. C. Arkin. *Motor Schema Based Mobile Robot Navigation*, The International Journal of Robotics Research, Vol. 8, No. 4, pp. 92-112, 1989.

[20] A. Saffiotti, Z. Wasik. *Using hierarchical fuzzy behaviours in the RoboCup domain*, Autonomous robotic systems: soft computing and hard computing methodologies and applications. pp. 235–262. Physica-Verlag GmbH. Heidelberg, Germany, 2003.

[21] S. Lenser, J. Bruce, M. Veloso. *A Modular Hierarchical Behaviour-Based Architecture*, Lecture Notes in Computer Science. RoboCup 2001: Robot Soccer World Cup V. pp. 79–99. Springer Berlin / Heidelberg, 2002.

[22] T. Rofer, H. Burkhard, O. von Stryk, U. Schwiegelshohn, T. Laue, M. Weber, M. Juengel, D. Gohring, J. Hoffmann, B. Altmeyer, T. Krause, M. Spranger, R. Brunn, M. Dassler, M. Kunz, T. Oberlies, M. Risler, M. Hebbela, W. Nistico, S. Czarnetzkia, T. Kerkhof, M. Meyer, C. Rohde, B. Schmitz, M. Wachter, T. Wegner, C. Zarges. *German team: Robocup 2005*. Technical report, Germany, 2005.

[23] A. Skarmeta, H. Martínez. *Fuzzy Logic Based Intelligent Agents for Reactive Navigation in Autonomous Systems*, Fitth International Conference on Fuzzy Theory and Technology, Raleigh (USA), 1997

[24] M. Loetzsch, M. Risler, and M. Jungel. *XABSL - A pragmatic approach to behaviour engineering*. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006), pp. 5124–5129, Beijing, October 2006.

[25] P. Kruchten. *Architectural Blueprints The "4+1" View Model of Software Architecture*, IEEE Software 12 (6), pp. 42–50, November 1995.

[26] B. W. Boehm. *Software Engineering Economics*, Prentice-Hall, 1981.

[27] N. Nilsson. *Teleo-Reactive Programs for Agent Control*, Journal of Artificial Intelligence Research, pp. 139–158, 1994.

Determining High Safety Risk Scenarios by Applying Context Information

Stewart Worrall, David Orchansky, Favio Masson, Juan Nieto and Eduardo Nebot

***Abstract*—When mining vehicle operators take risks, there is a increased probability of an accident that can cause injuries, fatalities and large financial costs to the mine operators. It can be assumed that the operators do not intentially take unnecessarily high risk, and that the risks are hidden due to factors such as adverse weather, fatigue, visual obstructions, boredom, etc. This paper examines the potential of measuring the risk of danger in a multi-agent situation by using the safe rules of operation defined by mining safety management.**

The problem with measuring safety is that the safe rules of operation are heavily dependent on the context of the situation. What is considered normal practice and safe in one part of the mine (such as performing a u-turn in a parking lot) is not safe on a haul road. To be able to measure safety, it is therefore necessary to understand the different context areas in a mine so that feedback of unsafe behaviour presented to the operator is relevant to the context of the situation. This paper presents a novel method for generating context area information using the vehicle trajectory information collected from a group of vehicles interacting in an area. Results are presented using real-life data collected from several operating fleets of mining vehicles. The algorithms presented have potential application to a large variety of environments including Intelligent Transportation Systems (ITS).

***Index Terms*—Vehicle Safety, Collision Avoidance, Context, Data Mining.**

I. Introduction

EACH year there are hundreds of mine haulage accidents that result in significant number of deaths, lost-time injuries and large financial costs due to machine downtime and the repair of equipment. The majority of these accidents occur as a result of a human operator failing to comprehend the risks involved when operating of the vehicle. This paper examines how high risk situations can be detected using rules of safe operation, and the importance of context in determining the level of risk. This allows feedback to be provided to the vehicle operator in the event a high risk situation, enabling the operator to alter their actions to reduce the risk.

An automated process for determining important context locations in a mine is presented, with results provided using real data collected from mining operations. This paper mainly refers to mining operations but the algorithms derived have application to all areas where diverse type of mobile agent interact within shared areas. These includes industrial applications such as stevedoring and construction where mixture of vehicles and people operate in a confined environment. It is also applicable to the Intelligent Transportation System area where we have high density of different type of agents such as vehicles, bikes, people operating in restricted areas such as road, intersections and parking lot.

Stewart Worrall, David Orchansky, Juan Nieto and Eduardo Nebot are with the Australian Center for Field Robotics at the University of Sydney. Favio Masson is with the Universidad Nacional del Sur. E-mail: s.worrall@acfr.usyd.edu.au

Fig. 1. (a) Dusty or (b) foggy conditions make it difficult for a vehicle operator to determine how close they are travelling to other vehicles.

Section II presents current research in this area, which focuses primarily on determining the likelihood of an collision occuring and warning the operator when the threat of collision is above some threshold. The problem with existing approaches is that it is often not possible to determine whether a situation will result in a collision with sufficient certainty (to minimise false alarms) until it is almost too late to avoid. This is considered to be a form of reactive feedback, and

it requires the operator to react quickly when an imminent collision is detected. An examples is presented in [1] where different level of warning are generated according to the proximity of the center or side of the road. Other approaches requires the sensing of location of other vehicles in proximity to generate some sort of alarm when some level of collision risk is evaluated. Furthermore, significant research to reduce false alarms has been done estimating roads and probabilistic methods to estimate time to collision with the evaluated roads [2], [3] [4].

An alternate approach presented in this paper is to provide preventitive feedback to the operators, meaning that the operator is warned when the risk of collision is considered high based on a predetermined set of safety rules. In the case of a mining environment, this includes rules governing minimum safe distances between vehicles, and speed limits for different areas of the mine. The problem is that the rules of safe operation change depending on the context of the situation. Context is a term introduced here to refer to the high level meaning of the situation that alters the rules used to determine risk. This can include concepts such as the semantics of the location, weather conditions, level of fatigue and many others. In a mining environment for example, vehicles on haul roads are required to maintain particular distance of separation which is greater than if they are located in a parking area or refuelling bay. In this example, a haul road is a different context area to the parking area and the refuelling bay. The paper focuses on the role of context areas in changing the rules of safe operation for a vehicle.

A digital map is required to determine the location of the roads and context areas. Determining the location of context areas can be done manually, though this is a time consuming process requiring skilled operators that know and understand the area very well. The mining environment presents a particular challenge for generating map information due to the dynamic nature of mining operations. Roads and open areas are constantly created and removed as the areas being mined change. These creates additional challenges to the operator since the environment and roads are changing all the time.

Sections III to V describe a novel technique for generating context area information by processing collected vehicle position information. Section III describes a system that automatically logs and collects vehicle trajectory information from a fleet of vehicles. Section IV examines techniques to filter this information to extract the road information. Section V presents a novel technique for extracting significant areas that can be used to form the context area map. Results showing the output of the algorithms using data from real-life mining operations is presented throughout the paper. Section VI provides discussion on the implementation. Finally, the conclusions are presented in Section VII.

II. A METRIC FOR SAFETY

Research has found that the 93% of vehicle accidents have human error as a causal factor [5]. In mining operations, safety managers attempt to minimise this risk of human error by designing protocols for safe operation, and training operators to follow these protocols. The safety rules evolve over time as the result of new research, and by analysing the cause of near misses and accidents. Despite the best efforts of mine management to implement rules and protocols for safe vehicle operation, accidents and near misses still occur. Vehicle operators do not normally take unacceptable risks knowningly. High risk situations can occur because of unknown risks caused by factors such as inexperience, adverse weather condition, lack of training, over confidence, boredom, fatigue and many others. Some examples of difficult environmental conditions from a typical mine are shown in Figure 1.

Fig. 2. Two areas of a mine that have different expected meaning for vehicle operators (different context meaning): (a) A dumping area where trucks park close together and (b) A haul road where vehicles must maintain separation of at least 50 m.

There are many groups currently researching Collision Avoidance Systems (CAS), which attempt to detect vehicles in the nearby area and estimate the risk of a collision (some examples include [11], [6], [7], [8]). Detection of the other vehicles can be either using sensors such as radar, stereo cameras, laser, etc, or using radio communication to broadcast the vehicle state wirelessly. In both cases, a warning is provided to the vehicle operator when the threat is determined to be above a certain threshold based on a variety of algorithms using the available vehicle state information of the nearby vehicles. The main issue with all of these approaches is that there is a trade-off between having a large number of false alarms by providing an early warning of a collision, and providing a warning of a collision at the last possible moment with fewer false alarms when a collision is almost unavoidable.

If the driver is presented with false alarms, it is likely that they will either not pay attention to the warnings or become annoyed at the unwanted distraction [9]. There are also clear benefits to providing warnings early to allow drivers more time to comprehend the situation [10]. This is a form of reactive feedback, which requires the operator to react fast enough to avoid the imminent collision.

An alternative strategy introduced in this paper is to use preventitive feedback in the case where a high risk scenario is detected, which is the precursor to a collision. The safe rules of operation are well defined for vehicle operations, including rules governing parameters such as minimum separation between vehicles and speed limits. By collecting vehicle state information from nearby vehicles, it is possible to detect breaches of these rules and provide feedback to the operator. This preempts a dangerous situation as the probability of a collision is highest when one (or more) of the safety rules has been breached. The feedback can then be provided to the operators allowing them time to process the information and modify their actions to minimise the risk in the situation.

The main difficulty in measuring safety is that safety is context dependent. The safe rules of operation vary depending on the circumstances of the situation which includes the location, speed, number of nearby vehicles as well as many other factors. The concept of a 'context area' is defined here as the human meaning attached to a specific area that defines the purpose of the area, consequently defining the expectation of behaviour in this area. A example of this can be seen in Figure 2, where two different context areas in a mine are illustrated. The top image shows a dumping area, where vehicles are queued close together waiting for the opportunity to unload the ore into a crushing machine. The bottom figure shows a haul road, where safety rules dictate that vehicles are not allowed to come within 50m of each other. The expectation of behaviour in the loading area is for close vehicle interactions at low speed, whereas in the haul road the vehicles travel at higher speed while maintaining more separation.

It is not feasible to provide feedback and warnings to the vehicle operators unless the context of the situation is taken into account. Presenting unnecessary warnings to an operator reduces the effectiveness of the system as the operator will be much less likely to trust the information, and will also likely be annoyed by the false information [9]. The remainder of this paper focuses on a novel method for extracting information about important context areas in a mine environment using data collected from vehicles in operation. These areas can be named by humans to provide the specific meaning, such as loading area or parking lot. These areas are then used to modify the rules as interpreted by the vehicle system to provide feedback to the operator in the event of a breach of these rules.

III. Collecting and Filtering Data

The mine environment is highly dynamic, with roads and areas being created and removed regularly. Mining roads are not sealed, and the colour and texture of the roads and traversible areas are often not distinguishable from the surrounding areas. In many roads or sections of the road there are no marks or berms to indicate the road boundaries. Furthermore the mine is a very dynamic process and roads are changing all the time. This means that image processing and aerial photography are not feasible to be used in the generation of up to date mine maps. The alternative method is to use vehicle position and trajectory information to generate the map.

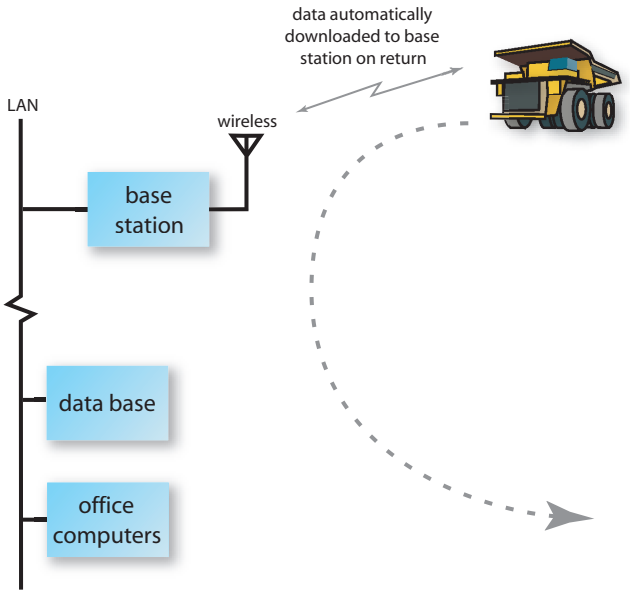

Fig. 3. Data is automatically collected from the vehicles when they are in range of a collection point (either directly or through the mesh).

A system has been designed and installed in several fleets of mining vehicles that can determine the vehicle state information, and to communicate this information with vehicles in the surrounding area using ad-hoc, mesh communication [12]. This system was designed to provide additional situation awareness to the vehicle operators, particularly in mining vehicles which often operate with poor visibility due to evnironmental conditions and visual obstructions from the vehicle cabin. Data is logged within each vehicle, including position information, heading, velocity and other vehicle state information. This data is collected when the vehicles move within range of a data collection point, either directly or though the mesh network of other vehicles. The data is downloaded and stored in a database as described in Figure 3, and can be used for creating mapping information.

The first step in the mapping process is clustering, which is necessary to reduce the amount of data and take the average of potentially noisy position data. Heading information must be considered as a separate dimension to allow the discrimination of position information from vehicles travelling in opposing directions on the same road. Position information from vehicles travelling in opposite directions often overlaps due to the vehicles not staying completely on one side of the road when overtaking or maneuvering around holes, etc. In addition, the position data is sourced from a GPS sensor which can potentially be noisy. These factors mean that the position information can be spread in the axis perpendicular to the direction of the road. To account for this variation, the clusters

must consider the heading in order to maintain separate lane information for each direction of travel.

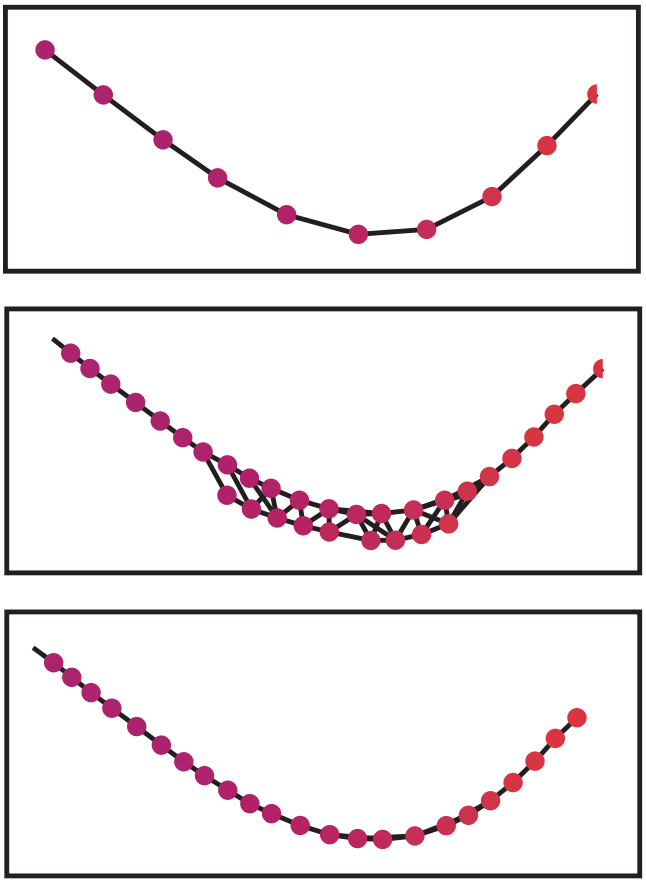

Fig. 4. Different cluster shapes used to group the GPS data (a) shows a curve with cluster size x=20m, y=20m, (b) shows cluster size x=7m, y=7m, (c) shows cluster size x=20m, y=7m.

The road width in mining can vary significantly, ranging from 10 to 30 or more meters in width. To provide an idea of the variety of scale of the vehicles involved, small vehicles can be around 2 meters wide, large vehicles can be over 7 meters in width. Tuning the parameters of a clustering algorithm to account for such variation is difficult, for example if points within a certain radius are clustered together then the radius required would need to be at least 20 to 30 meters to account for the maximum road width. Increasing the collection radius for a cluster means that the clusters are spread further apart along the axis of the road and some fidelity of the road shape is lost, especially around corners and at intersections. A solution to this problem is to change the shape of the area used to collect the position information into a cluster. Figure 4 shows the outcome when using a rectangular shaped cluster area. Points are incorporated into a cluster differently in the x axis (perpendicular to the direction of travel) compared with the y-axis (following the direction of travel. If points are added into a cluster from a larger range in the x-axis, the fidelity of the road shape is better maintained as seen in Figure 4(c). Figure 4(b) shows the output if the clusters are too small, leading to multiple lines of clusters representing the same section of road.

IV. Extracting Roads

Once the point clusters are generated, the next step is to determine the relationship between each cluster. This process involves linking the clusters, and is used to form the road information. A road is defined here as a linked set of clusters where the motion of the vehicle is constrained to a single dimension (direction). This means that each cluster is linked forward to a single cluster, and back to a single cluster. This paper describes three methods of linking clusters that contain position and heading information.

The above definition of roads and areas allows the mine layout to be represented as a directional graph, or digraph. The context areas are used to make the verticies of the graph and the roads form the directional edges that join the verticies. It is possible to form the graph representation by defining a road as the single directional path between two context areas, which can be intersections, or other context areas. This formal representation of this graph structure is presented in Equation 1, and the graph structure can be seen in the roads and areas shown in Figure 8.

The roads are extracted from the cluster data as places where normal motion is constrained to a single direction of travel. Once these areas are determined, the remaining clusters can be used form areas where the direction of travel is less constrained, where vehicles can move in different directions. These areas represent increased likelihood of an important area with contextual meaning as there is a large potential range of maneuvers in these areas.

$$\begin{aligned} \text{Graph of Mine } G &= (V, E) \\ \text{vertices } V &= v_1 \cup v_2 \cup \cdots \cup v_n \\ &= \text{Set of } n \text{ Context Areas} \\ \text{edges } E &= e_{1,2} \cup e_{2,1} \cup \ldots etc \\ &= \text{Set of Roads} \end{aligned} \tag{1}$$

where

$e_{i,j}$ = directional link between area i and j

A. Geometric Relationship

The road information can be determined by considering the geometric relationship between the clusters. Equation 2 introduces a metric that is calculated using a function of the distance between clusters (ΔS), bearing of the other cluster relative to the test cluster heading (ϕ), and the difference in heading between the clusters ($\Delta\theta$). Each of these measures are weighted using the parameters α, β and γ and added to give the metric value. These parameters are tuned to suit the application, though there is scope for future work in using feedback to improve the parameter selection within a probabilistic framework.

$$\text{metric } = \alpha\Delta\text{S} + \beta\phi + \gamma\Delta\theta \tag{2}$$

The metric is calculated for each cluster relative to the other nearby surrounding clusters. When the metric is within a given

threshold, the link is created between the clusters. This strategy was successful in most scenarios, with the exception being if two unconnected roads are too close together for the selected distance metric. In this case, there is a tendancy for incorrect links to be made as illustrated in Figure 5(a).

B. Temporal Relationship

The second approach for linking clusters considered in this paper involves fitting the logged vehicle trajectories to the clusters. A link is strengthened when a vehicle trajectory passes between two clusters. The strength of the link is measured by the number of times a vehicle moves between the clusters. A threshold is used to determine what strength of link is required to create the road, allowing this process to filter out incorrect links. Using vehicle trajectories to generate the links provides a robust linking method that is less likely to fail when different roads are close together. This is because the links are only created between clusters where vehicles have actually traveled. Part (b) of Figure 5 shows the result of the linking process using vehicle trajectories. On the roads, the two approaches for linking clusters produces similar results. The main benefit of the vehicle trajectory approach is seen around intersections.

C. α-Shape

The underlying shape of a set of points can be determined by using the alpha shape technique [13]. This technique uses a circle of radius α to essentially 'scoop' out the area surrounding the border of the points. The process begins by taking a set of points to be included in the shape, and determining the circumcircles that can be created using each pair of points. The boundary of the shape is defined by all the circumcircles that do not contain one of the other points in the set. This means that an α-shape can contain holes and concave areas. As α approaches infinity, the shape approaches the convex hull. The value of α was selected to be twice the size of the distance between clusters, meaning that all clusters that should be joined are within the range of the nearest points when creating the circumcircles.

Figure 6(a) shows the set of clusters used to make the α-shape. Figure 6(b) shows the resulting α-shape, with the circumcircles drawn when they do not contain a different point from the remaining set. It can be seen in this shape that when the roads are far apart, they are considered as separate lines both circumcircles created from the pair of points do not contain another point in the set. When the roads are closer together, the inner circumcircles contain points from the opposing side of the road to create a filled in shape (indicated as a black filled polygon in the image).

To provide the separation between roads, it is possible to consider the heading as an additional dimension to the α-shape algorithm. In this case, the test of other points that lie within the circumcircle excludes points that have a heading difference that is larger than a particular threshold. This means that the clusters that belong to the opposing side of the road are not included, and the α-shape can represent the different sides of the roads as shown in Figure 6(c).

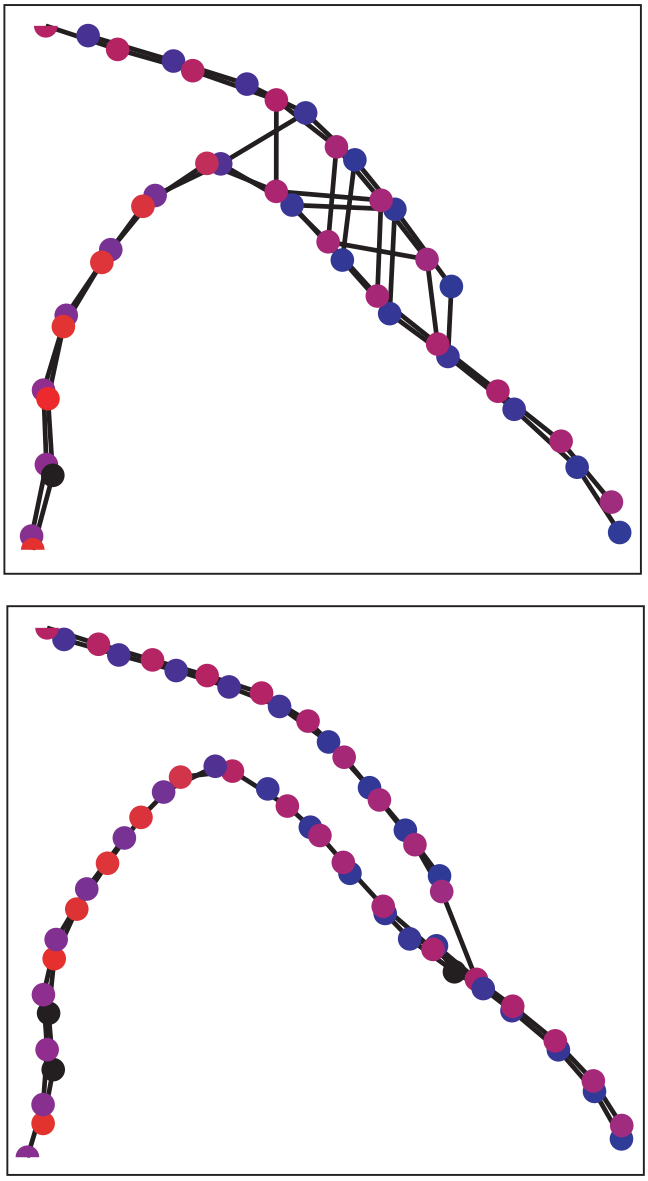

Fig. 5. Shows a potential issue with geometric based linking of clusters (a) geometric linking where incorrect links are made to a nearby road (b) linking using the actual vehicle trajectories solves this problem.

V. Context Area Shapes

Section IV described the process of determining the location of roads within the mine where the vehicle motion is constrained to a single direction. The remaining clusters then indicate the areas where the vehicle movement is less constrained and more complex than on a road. The additional complexity generally means there is a greater chance of a misunderstanding between two nearby vehicles, and that the area has some additional meaning to the vehicle operators. These areas include intersections and other open areas such as parking lots, loading areas, dumping areas, refuelling bays, etc. These areas are of interest as the expectation of behaviour of other nearby vehicles is likely to be different to being on a road. In a parking lot for example, there is an expectation that a vehicle is likely to drive slower and maneuver into a parking space at any moment.

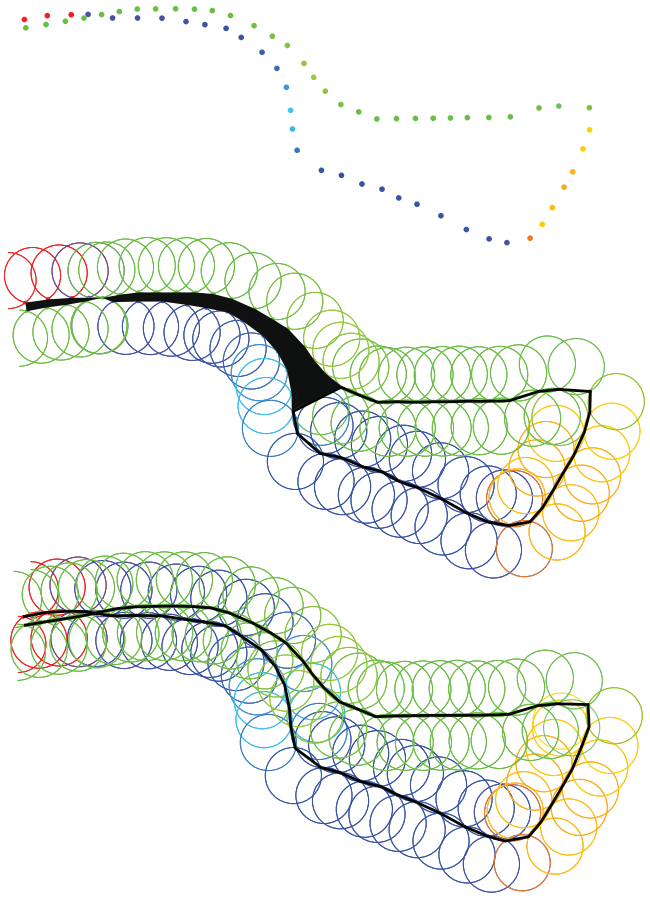

Fig. 6. Using α-shapes to extract road information from a set of clusters. (a) shows the clusters only, (b) shows the α-shape with the circumcircles not containing another point from the set drawn and (c) shows the α-shape created when the circumcircle test ignores the points where the difference in heading is greater than a threshold (in this example, 90 degrees).

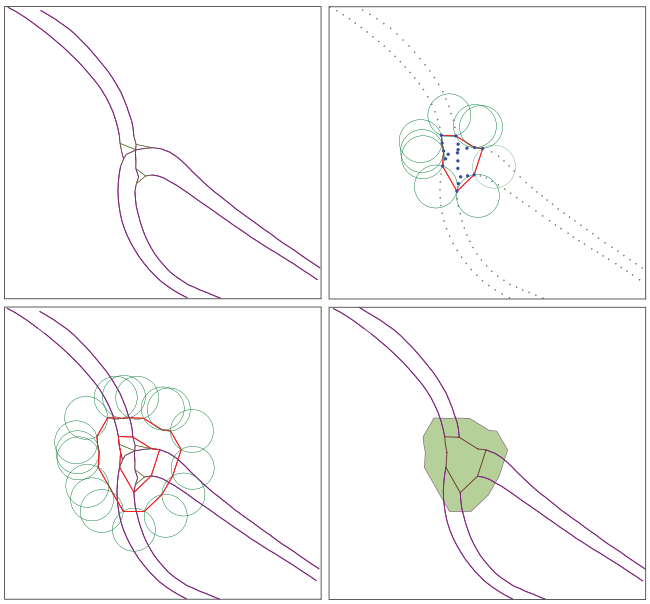

Fig. 7. Using α-shapes to build an intersection. (a - top left) shows the roads created by the process described in Section IV-C. (b - top right) shows the clusters not associated with a road used to make a normal α-shape. (c - bottom left) The points in the same intersection are dialated using a hexagon shape of points. (d - bottom right) The final context area is shown.

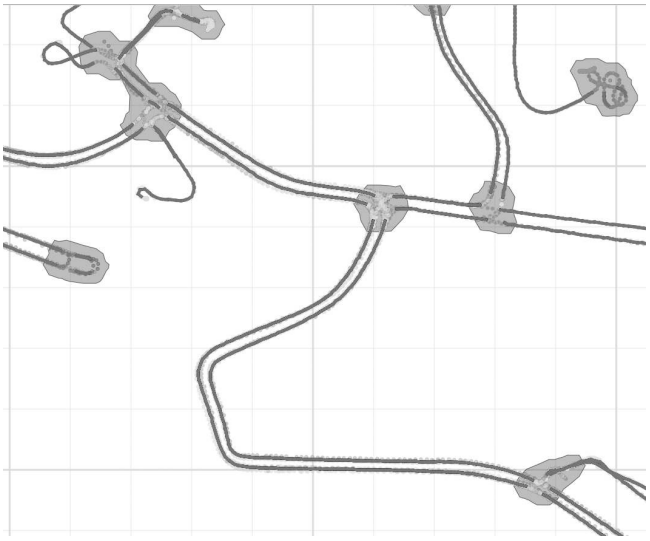

Fig. 8. The map shows the output of the algorithm to generate the roads and determine important areas. The roads are represented by black lines, the areas by filled polygons. Position information used to generate the map is plotted behind the roads.

The context area is defined by creating α-shapes out of the clusters that are not already affiliated with a road. Figure 7(a) shows a typical intersection in a mine, with the roads overlaid. At the intersection, the roads finish because they link forward to more than one other cluster at the junctions. Figure 7(b) shows the clusters only, the clusters not associated with a road form a shape using the α-shape algorithm.

At this point, the shape represents the intersection, but too tightly because each cluster represents only the average point from the data collected in the cluster shape (described in Section III). The area is better represented by dialating the clusters in the context area and using the expanded set of points to make a larger shape. The clusters are dialated by the normal distance between clusters, which was empirically set to 10 meters. Six points were taken at this radius around each point in the context area making a hexagon shape, with the intention that this would be sufficient to represent the expanded shape. The resulting points and α shape can be seen in Figure 7(c). The final context area shape can be seen in 7(d).

Figure 8 shows the combination of many context areas and roads that was created from real data collected from a mining operation. The roads are shown as black lines, and the areas are represented by filled polygons. The GPS information that was used to create the roads is plotted behind the roads to show how well the context areas cover the area used by the vehicles.

VI. Experimental Results

Figure 9 shows the roads and context areas created using the algorithms described in this paper overlaid on an aerial photograph of the mine. The high risk areas in this image are the intersections, especially the large, four-way intersection in the center of the image. This intersection is a potentially high risk environment as there could be vehicle approaching from any of four directions, and the intended destination of each

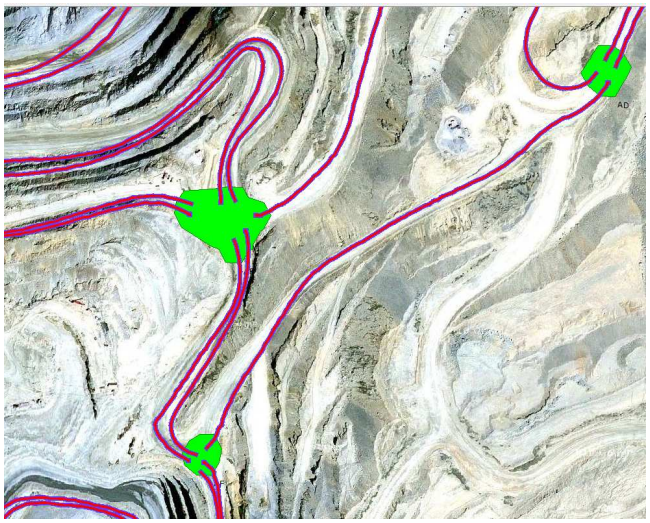

Fig. 9. Roads and context areas are overlayed on a aerial image of the mine site. The important context areas in this image are the intersections, particularly the large, four-way intersection in the center of the image.

Fig. 10. The operator interface provides situation awareness information to the driver. There are two vehicles displayed on the screen that outside the operator's line of sight. An area of interest is shown at the top of the screen representing a large intersection.

vehicle is unknown to the other vehicles. These roads and context areas are then automatically distributed to the fleet of vehicles and are used for the operator interface.

Figure 8 shows the quality of the areas generated from the algorithms introduced in this paper. The correctness of the areas can only be determined by the mine personnel who are familiar with the roads and layout of the mine, and is essential before the maps can be used in the operator interface. The feedback from the mine personnel is that the algorithm provides a good representation of the context areas in the majority of cases. This subjective analysis is not yet quantified. Future research in this area will examine metrics to automatically determine the quality of the generated maps. Two main factors were determined that impact the quality of the created map. In the event of poor GPS reception, it is possible that the noise in the position estimate can cause errors in the context area shape. In addition, data collected from vehicles performing maneuvers that are uncharacteristic given their location presents another potential cause of incorrectly generated context areas. An example of this is if a vehicle performs a u-turn maneuver on a section of haul road (which they are not supposed to do), this area could potentially be incorrectly interpreted as being a context area. Detecting and filtering out these cases is an area of future work.

Figure 11 shows the operator interface in the vehicle cabin designed to improve the operator's situation awareness. At a glance, it is possible for the operator to determine that they are in a context, or important area (labelled in this case as a Crusher Area). The nearby vehicles are displayed, and the distance icon accompanied by a short sound indicates that the distance is too close to the vehicle in front based on the rules determined for that specific context area. When the rules implemented for this specific context area are breached, the icons shown on the bottom of the image are displayed along with the appropriate sound warning.

Figure 10 shows the interface working in a mine site, providing situation awareness to the operator. The screen shows two vehicles in front, neither of which can be seen directly from the vehicle. The coloured area at the top of the display indicates that there is an important area (in this case an intersection) around the next corner which is not in a line of sight from the vehicle. This simplified representation of the environment allows the operator at a glance to improve their understanding of the current situation.

The current vehicle position relative to the roads and areas on the map provide an important piece of information to the vehicle operator. In poor weather conditions, it can be difficult to determine the vehicle location relative to the intersections and other areas, especially for an inexperienced driver. Providing the map with hilighted intersections and areas to the operator should improve the awareness of the current location.

Fig. 11. Display implementation currently installed in mining vehicles. The context area name can be seen in the bottom left of screen. Several warning icons representing breaches of the rules are presented.

VII. Conclusion

This paper introduced a method of determining risk using context based rules. A method for generating up to date road maps including the evaluation of context areas was presented using vehicle trajectory data collected from real-life mining vehicle fleets. The context area was generated using a process of clustering raw data, extracting road information from the relationships between the clusters and finally using the remaining clusters to generate an α-shape representing a potential area of interest. The context areas can be provided with a label by a human to make it more understandable to the vehicle operator. There is scope for future work in this area to automate this process by probabilistically determining the semantic meaning of each area.

Without the context information, it is not possible to measure risk based on rules that vary depending on location. It is necessary to consider the context of the situation before providing potentially incorrect feedback to the vehicle operator. Unnecessary or false alarms are detrimental to the trust and usefulness of the system, and can potentially be distracting and annoying. This work can be extended outside the field of mining operations. There are many parallels with the operation of mining vehicles and passenger vehicles. Mining operations are generally present a much more difficult environment with unsealed roads, large range of vehicle sizes and 24 hour operation.

References

[1] E. Nebot, J. Guivant, and S. Worrall, "Haul truck alignment monitoring and operator warning system," *Journal of Field Robotics*, vol. 23, no. 2, pp. 141–161, March 2006.

[2] S. Worrall and E. Nebot, "Using non-parametric filters and sparse observations to localise a fleet of mining vehicles," in *IEEE International Conference on Robotics and Automation, 2007*, April 2007, pp. 509–516.

[3] S. Worrall and E. Nebot, "A probabilistic method for detecting impending vehicle interactions," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 19-23 2008, pp. 1787 –1791.

[4] G. Agamennoni, J. Nieto, and E. Nebot, "Mining gps data for extracting significant places," in *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 1860–1867.

[5] M. S. Sanders and B. E. Shaw, "Research to determine the contribution of system factors in the occurrence of underground injury accidents," USBM OFR 26-89, NTIS PB 89-219638/AS, 165 pp., Tech. Rep., 1988.

[6] Y.-J. Chen, C.-C. Chen, S.-N. Wang, H.-E. Lin, and R. Hsu, "Gpsensecar - a collision avoidance support system using real-time gps data in a mobile vehicular network," in *Systems and Networks Communication, 2006. ICSNC '06. International Conference on*, 2006, pp. 71–76.

[7] J. Huang and H.-S. Tan, "Impact of communication reliability on a cooperative collision warning system," *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pp. 355–360, September 2007.

[8] O. Gietelink, D. Verburg, K. Labibes, and A. Oostendorp, "Pre-crash system validation with prescan and vehil," *Intelligent Vehicles Symposium, 2004 IEEE*, pp. 913–918, June 2004.

[9] B. Donmez, L. N. Boyle, J. D. Lee, and D. V. McGehee, "Drivers' attitudes toward imperfect distraction mitigation strategies," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 9, no. 6, pp. 387–398, Nov. 2006. [Online]. Available: www.sciencedirect.com/science/article/B6VN8-4JJ2BVX-1/2/a091f0f1acc2864af9b5d9b66b33681e

[10] J. D. Lee, M. L. Ries, D. V. McGehee, and T. L. Brown, "Can collision warning systems mitigate distraction due to in-vehicle devices?" NHTSA, Tech. Rep., May 2000. [Online]. Available: www-nrd.nhtsa.dot.gov/departments/nrd-13/driver-distraction/PDF/31.PDF

[11] S. Kodagoda, S. Sehestedt, A. Alempijevic, Z. Zhang, A. Donikian, and G. Dissanayake. Towards an enhanced driver situation awareness system. *Industrial and Information Systems, 2007. ICIIS 2007. International Conference on*, pages 295–300, Aug. 2007.

[12] S. Worrall, D. Orchansky, and E. Nebot, "Improving situation awareness with a high integrity proximity system," *Australasian Mine Safety Journal*, vol. 2, pp. 94–97, 2009.

[13] H. Edelsbrunner and E. Mücke, "Three-dimensional alpha shapes," in *Proceedings of the 1992 workshop on Volume visualization*. ACM, 1992, p. 82.

Probabilistic Estimation of Unmarked Roads using Radar

Juan I. Nieto, Andres Hernandez-Gutierrez and Eduardo Nebot

***Abstract*—This paper presents a probabilistic framework for unmarked roads estimation using radar sensors. The algorithm models the sensor likelihood function as a Gaussian mixture model. This sensor likelihood is used in a Bayesian approach to estimate the road edges probability distribution. A particle filter is used as the fusion mechanism to obtain posterior estimates of the road's parameters. The main applications of the approach presented are autonomous navigation and driver assistance. The use of radar permits the system to work even under difficult environmental conditions. Experimental results with data acquired in a mine environment are presented. By using a GPS mounted on the test vehicle, the algorithm outcome is registered with a satellite image of the experimental place. The registration allows to perform a qualitative analysis of the algorithm results. The results show the effectiveness of the algorithm presented.**

***Index Terms*—Localisation, radar sensors, particle filter**

I. Introduction

ONE of the most important tasks in autonomous navigation is the segmentation of the road course [1], [2]. The design of a robust road estimation system needs to consider variations among roads such as paved or unpaved, marked or unmarked. Roads in urban environments have in general well defined lane marks, constant width and well defined corners. On the other hand, roads in rural areas tend to have much less structure and are in general unmarked. Mining roads are a good example of the later. Roads in mines are unpaved, no lane markers are presented, and a constant variation of the road width can be found. The lack of structure in rural areas makes the vehicle relative positioning process harder.

Road estimation is not only important for autonomous navigation but it can also be utilised for driver assistance systems [3]. An example of a warning system aiming driver assistance in unmarked roads is presented in [4]. The system estimates the relative vehicle position using range information gathered from a laser scanner. The range data is used to measure the relative distance between the vehicle and poles located at the side of the road. The additional infrastructure added on side of the road ensures the system can reliably detect when a vehicle gets too close to the center or side of the road. The main disadvantage of the system is the need for installation and maintenance of infrastructure (poles at the side of the roads).

The design of a road estimation system will also depend on the sensors selected. Radar sensors have proven to be very robust to different weather conditions such as dust, fog and snow. These positive aspects have encouraged car manufactures to equip vehicles with radar sensors to perform intelligent tasks, such as collision avoidance, traffic scene interpretation and moving objects detection. However, radar based navigation systems are seldom used to provide information related to vehicle position relative to road edges. This is due to the cost of the technology and also partly to the difficulty in processing the information returned. For example, due to the wider beam pattern, radar information from a given bearing will in general consist of various returns at different ranges. At the same time objects with large radar target that are not within the nominal current bearing cone could potentially generate an erroneous return at the present bearing. These issues with radar add an extra complexity and therefore this sensor modality is usually avoided when not specifically needed. In most applications vehicles are equipped with other sensors modalities for road shape estimation, such as cameras and laser scanners. For instance, vision systems are well suited to detect lane markings. Laser scanners have high resolution in azimuth and wide field of view which makes them suitable for the extraction of berms and curves along the edges of the road. The main disadvantage of visual and laser sensors is the lack of robustness against bad weather conditions such as fog and dust for example.

The work presented in this paper aims to develop a system to work in unpaved and unmarked roads and under harsh environmental conditions. This is the case for example of autonomous operation in a mine where the system needs to operate reliable 24/7 under all environmental conditions. The harsh conditions requirement restricts the choices of possible sensors, being radar the most suitable one. For this type of applications, the radar becomes an essential component to enable automation and operator aiding. The algorithm presented in this work performs Bayesian estimation of the road's edges which allows to obtain the probability distributions of the road parameters and vehicle relative position. The system is robust to sensor noise since incorporates the uncertainty in a Gaussian mixture likelihood function.

This paper is organized as follows. Section II presents related work. An overview of the algorithm is described in section III. Section IV shows the model to represent the road. The algorithm proposed using particle filters is explained in section V. Information related to the radar sensors as well as GPS positioning is detailed in Section VI followed by experimental results in Section VII. Conclusions and future work are presented in Section VIII.

Juan I. Nieto, Andres Hernandez-Gutierrez and Eduardo Nebot are with the Australian Center for Field Robotics at the University of Sydney. E-mail: {j.nieto,a.hernandez,nebot}@acfr.usyd.edu.au

II. Related Work

Several approaches for road tracking have been proposed during the last two decades. These works have emerged as a need of intelligent transportation systems for an everyday crowded world. Vision-based road tracking systems are presented in [5], [6] and [7]. The system described in [5] estimates the curvature and orientation of a road based on monocular frames and a model of the road. First, the algorithm segments the acquired images to extract road edges and orientation. In order to remove outliers, the least median squares filter is applied. Using inverse perspective projection, feature points in the image are projected back onto the ground plane and the coordinates of the marking lanes, relative to the camera are computed. This technique works efficiently for marked roads. In contrast, [7] presents an approach that deals with both marked and unmarked roads using a monochromatic camera. Applying a statistical model, this method copes with problems such as occlusion and not well defined marks. Besides the lane detection and tracking estimation, this algorithm also estimates the vehicle position within the road. The approach presented in [8] is based on the extended Kalman filter and a linear model to estimate and track successively the road curbs. This method works well when curbs are well defined.

In addition to mono-sensor approaches, algorithms that improve robustness by fusing information from different sensors modalities have been proposed in [9] and [10]. In [9], the detection of curbs and walls are based on a laser scanner, a GPS device, an inertial measurement unit (IMU) and a monochromatic camera. This system is capable of detecting curbs alongside the vehicle and at the front. Our work is more related to the framework shown in [10], where the authors present a technique that integrates information gathered by optical and radar imaging sensors. By using deformable template models of lane and pavement boundaries together with likelihood models of the sensors, the detection and tracking of lane and pavement boundaries are implemented in a Bayesian framework. The work presented here is based on radar only.

III. Algorithm Overview

In this paper we present a probabilistic road estimation system to evaluate both the width of the road, and the vehicle position with respect to the road's edges. The proposed method is based on information gathered by a millimetre-wave radar that detects salient points located around the vehicle. The sensor's returns are segmented into different intensity levels. Sensor data is then modeled as a Gaussian Mixture Model (GMM) as the uncertainty associated to each intensity level of each particular radar bearing return is modeled as a Gaussian. Monte-Carlo sampling is used to get *a-priori* estimates of the road. The system uses Clothoid functions defined over a set of parameters to model the road shape. Once road samples are obtained, they are fused with the likelihood GMM built with the observations. The fusion is done using a particle filter, which weights the road samples according to its probabilities. As a result, the system provides a stochastic road's shape model. Figure 1 presents a block diagram of the framework.

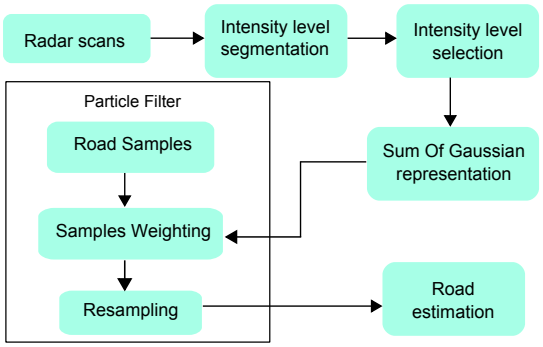

Fig. 1. Block diagram of the proposed method to estimate the shape of the road using radar information.

IV. Road Model

Clothoids functions are widely used for road design in order to connect geometry between a straight line and a circular curve. The main characteristic of a Clothoid curve is that the curvature is proportional to the inverse of the radius. The Clothoid's geometry models well the centripetal acceleration experienced by a vehicle approaching a curve.

To model the road ahead of the vehicle, we use a polynomial cubic approximation of a Clothoid [11]

$$y(x) = y_0 + tan(\phi)\ x + C_0 \frac{x^2}{2} + C_1 \frac{x^3}{6} \qquad (1)$$

where y is the distance between the vehicle and the proposed left road, being y_0 the initial offset. The orientation angle of the road relative to the vehicle coordinate frame is given by ϕ. C_0 and C_1 represent the curvature and curvature rate respectively. Figure 2 illustrates the model described in Equation 1. Although the width of the road W is not included in Equation 1, this parameter will be also estimated as it will be shown in Section V-A.

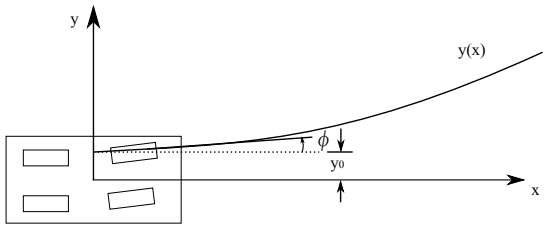

Fig. 2. Model of the road using a Clothoid function.

V. Road Estimation using Particle Filters

Particle filters are a recursive Bayesian estimation tool based on Monte Carlo sampling methods. Unlike the Kalman filter, particle filters have the advantage of not being subjects to any linearity or Gaussianity constraint on both the transition and observation model [12], [13]. The basic idea behind particle filters is to represent probability distributions by a set of random samples or particles. In order to make inference about a dynamic process, particle filters usually use two models; a state evolution model and an observation model, being both

models perturbed by noise. The filter is articulated through two main steps, prediction and update. During the prediction stage, particles representing random samples of the estimated states are passed through the system model. During the update stage, the new measurements received are used to update the likelihood of the prior samples and obtain normalised weights for the samples. The weights represent the likelihood of the model encapsulated by the individual particles [14].

The particle filter implemented here is the Condensation algorithm, presented in [15] for tracking curves in visual clutter. The state vector is represented by the parameters used to model the Clothoid, shown in Equation 1 plus the road width W.

$$\mathbf{x} = [Y,\ \phi,\ C_0,\ C_1, W]^T \tag{2}$$

A. Process Model

The vehicle model is approximated by the Ackerman bicycle model

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} v(t)\ cos\theta \\ v(t)\ sin\theta \end{bmatrix} \tag{3}$$

Fusing Equation 1, which represents the shape of the road, with the vehicle model described in Equation 3, the state evolution model can be obtained.

$$\begin{bmatrix} Y_{t+1} \\ \theta_{t+1} \\ C_{0_{t+1}} \\ C_{1_{t+1}} \\ W_{t+1} \end{bmatrix} = A(\Delta s) \begin{bmatrix} Y_t \\ \theta_t \\ C_{0_t} \\ C_{1_t} \\ W_t \end{bmatrix} + \begin{bmatrix} 0 \\ -\Delta\theta_t \\ 0 \\ 0 \\ 0 \end{bmatrix} + \mathbf{w(t)} \tag{4}$$

$$A(\Delta\ s) = \begin{bmatrix} 1 & \Delta s & \frac{\Delta s^2}{2} & \frac{\Delta s^3}{6} & 0 \\ 0 & 1 & \Delta s & \frac{\Delta s^2}{2} & 0 \\ 0 & 0 & 1 & \Delta s & 0 \\ 0 & 0 & 0 & 0.99 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

where Y represents the distance between the vehicle and the proposed road. ϕ is the orientation angle of the road with respect to the vehicle location. C_0, C_1 and W are the curvature, rate of curvature and width of the proposed road. Δs represents the longitudinal displacement along the curve from time t to $t+1$ and **w(t)** the process noise, which is a zero-mean white noise sequence. The vehicle speed involved in (3) as well as its longitudinal displacement, which is represented by Δs in (5), are provided by the GPS receiver. Figure 3 shows an example of road samples superimposed on a satellite image.

B. Sensor Likelihood Function

We assume the sensor possesses normally distributed noise. Thus, a radar scan can be represented as a Gaussian Mixture Model (GMM). Radar returns are originally in polar coordinates, being described by range r and bearing θ information. For the road estimation process, data is converted from polar to Cartesian coordinates that is the space we are interested.

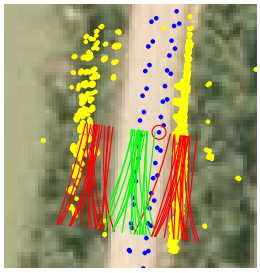

Fig. 3. The red curves in the figure illustrates the road samples using the Clothoids model. Yellow points denote the selected radar returns and blue dots the GPS trajectory.

$$f_{x,y}(r,\theta) = \begin{pmatrix} r\ cos\theta \\ r\ sin\theta \end{pmatrix} \tag{6}$$

The sensor information is considered to be uncorrelated in the original polar space, therefore the covariance matrix associated to a radar return can be represented as

$$\Sigma_{r,\theta} = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{pmatrix} \tag{7}$$

The covariance matrix in Cartesian coordinates is obtained through a linearisation of Equation 6.

$$\Sigma_{x,y} = J\ \Sigma_{r,\theta}\ J^T \tag{8}$$

where J is the Jacobian of $f_{x,y}(r,\theta)$ and $\Sigma_{r,\theta}$ is the covariance matrix associated to the radar data as in (7). Figure 4 depicts an example of a GMM representing a radar scan. As seen in this figure, the edges of the road can be well defined by the Gaussians.

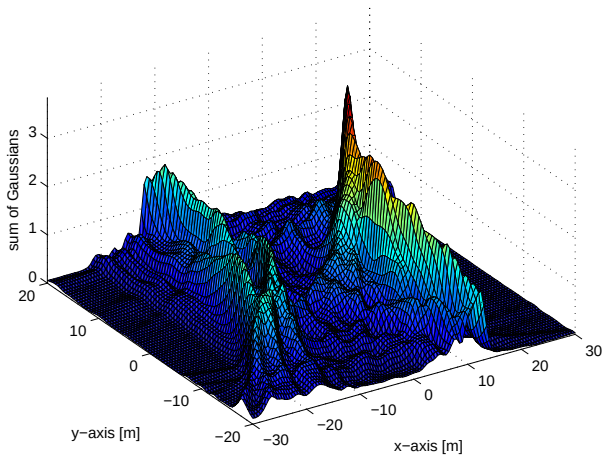

Fig. 4. GMM representing a radar scan.

C. Model Update

The prediction stage gives a set of particles, each representing a different set of road parameters. In order to decide which proposed road better describes the shape of the road, we use an objective function to weight each particle according to the

likelihood obtained using the GMM sensor model. Firstly, the left and right proposed roads are discretised and evaluated in the sum of Gaussian map as follows

$$cl_i^s = \sum_{k \in P} L_k \tag{9}$$

$$cr_i^s = \sum_{k \in P} R_k \tag{10}$$

where L_k and R_k are the Gaussian values along the left and right proposed roads. The total cost for a proposed road, $x_i^s \in X_s$, is given by the sum of these individual distances as follows

$$C_i^s = cl_i^s + cr_i^s \tag{11}$$

This cost value will be used by the particle filter to weight each particle.

Using the total cost associated to each of the proposed roads, the particles are weighted by applying the following expression

$$w_{i_t} = \frac{C_i^s}{\sum_{i \in S} C_i^s} \tag{12}$$

where w_{i_t} is the ith weight of the proposed road at time t, and N the number of particles.

D. Resampling

An important drawback of particle filters is the so called *degeneracy problem*. The degeneracy problem is when most particles have weights very close to zero therefore only few samples are being used to represent the distribution. Resampling is used to avoid this problem by selecting and multiplying the most "important" particles (biggest weights) and eliminating the particles with the lowest weights. In other words, the resampling process selects the set of particles that best represents the distribution. The filter implemented here is a sampling importance resampling (SIR) as presented in [14] and [15] and uses Stratified resampling [16].

VI. Sensors

The main objective of road estimation is vehicle relative positioning and so global position estimation is not needed. However, due to the difficulties arising in attempting to get ground truth of the relative position to the road, a quantitative analysis of the results is not possible and instead we provide a qualitative analysis by superimposing the vehicle trajectory and estimated road with a satellite image. A GPS sensor mounted on the experimental vehicle is used to get position estimation. The data was gathered in a mine, with the sensors mounted on a truck. Sensors' details are presented next.

A. GPS Data

The GPS used for this work is a standard A12 from Thales Navigation, working in autonomous mode which gives and accuracy between 2 and $10m$. Information such as latitude, longitude, heading and speed of the vehicle is available every second.

B. Radar Data

External information is acquired by a radar scanner working at 94 GHz [17]. The radar has an uncertainty of $20cm$ in range and 1 deg in angle. The radar returns different intensity values. In order to convert from intensity to range, we use a linear model

$$r = Ab - C \tag{13}$$

where r is the range from the radar return, A represents the radar resolution, that in this case is 23.52 $cm\ bin^{-1}$, b is the bin number for the return and C is the offset between the transmitter and the receiver in the radar, which is 60 cm for this sensor. Figure 5 shows an example of intensity return versus bin values for a particular angle. This radar scan was acquired at 33 deg. In this case, the maximum intensity value, $94.99dB$, corresponds to an object located at bin 63; therefore, the range to this object is 21.744 m.

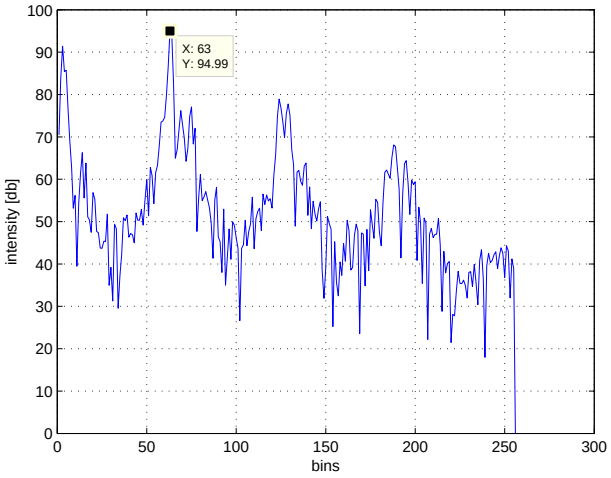

Fig. 5. Reflectivity values scattered by objects located at a given rotation angle of the radar. Range is discretized into bins with 0.2353 m resolution. For an object found at bin 63 having an intensity of $94.99dB$, its range will be 21.744 m.

For the work presented, we limited the radar data to 200 bins, that represents a maximum range of approximately 46 m. This range was enough to detect the road boundaries. Because we are interested in detecting the edges which in general reflect large intensity values, we use thresholds to select the largest intensity values and segment each return into a discrete number of ranges. The different intensity values correspond to reflections from objects that are at different distances to the sensor. In the results presented here we use thresholds at $51dB$, $71dB$, $76dB$, $80dB$ and $82dB$. These values were obtained through experimentation. It is important to note that the segmentation is mainly used to reduce the amount of data. We did not find the threshold values to be an important factor in our implementation as long as the values with the highest intensities were included.

Figure 6 depicts an example of the segmentation process, superimposed on a satellite image. Objects represented by blue and green points reflect back an intensity value greater than

the first and second thresholds; whereas objects represented by yellow, red and violet points have an intensity value greater than the third, fourth and fifth thresholds respectively. In order to filter out radar information corresponding to the roof of the vehicle, data located within a range of $2.5\ m$ is removed. This is represented in the figure with a small circle around the sensor position.

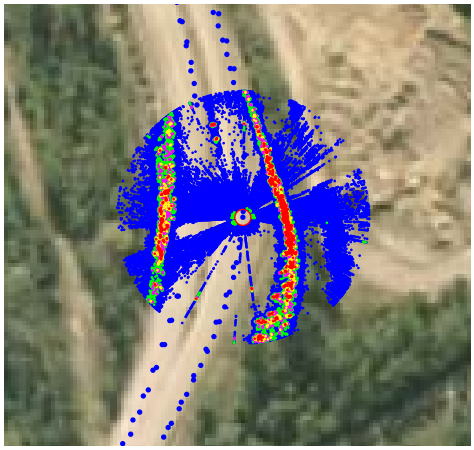

Fig. 6. Reflectivity values segmented into five different levels. Intensity values represented by green points are considered in the road shape estimation; whereas, points in blue are removed. GPS position is represented by blue dots along the trajectory.

VII. Experimental Results

The experimental data was collected in a mine, in southeast Queensland, Australia. A millimetre radar (94 GHz) was mounted on top of the experimental vehicle. The whole vehicle trajectory was around $1.5\ km$.

Figure 7 displays an image of the experimental area, together with the vehicle trajectory, radar returns and road estimates. Yellow points represent the targets detected by the radar sensor along the trajectory. As can be seen in the figure, high intensity values are returned not only from the edges of the road, but also from the trees, obstacles located around the path and also there are high intensity returns from within the road. The large amount of data and the noise included in the sensor returns make the road estimation a very difficult problem. The vehicle position reported by the GPS device is illustrated in blue dots within the path.

The red lines showed in Figure 7 show the final left and right road edges estimated. The filter was run using 500 particles. As shown in Figure 7, the estimated edges define appropriately the edges of the road along the trajectory. Note that part of the offset observed with the satellite image is due to the error in GPS position which is used to register the radar points with the image.

Figures 8 and 9 show a zoom-in of the estimated road edges together with their uncertainty obtained from the particle filters. The dashed lines represent the $\pm 2\sigma$ uncertainty obtained from the samples variance. As seen in Figure 8, the actual road's edges are found within the uncertainty area. Note that the uncertainty in the first curve (Figure 8 bottom part), is bigger than the one near the bridge (Figure 8 top-left). This

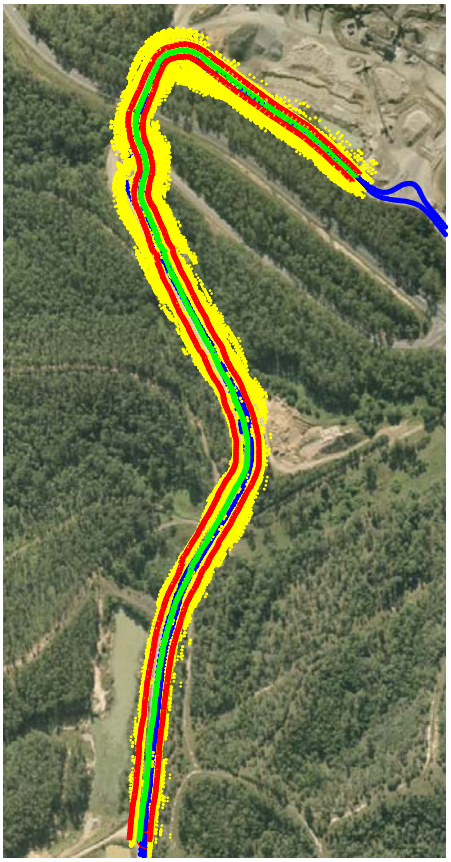

Fig. 7. Estimated edges of the road and radar data superimposed on the satellite view. The edges and the centre-line of the road are represented by the red and green line respectively; whereas, the vehicle position is illustrated by blue dots along the trajectory.

is because the area near the bridge presents a higher density of points, which gives a better road estimate. See also the area around the roundabout, where the edges are very poorly defined. Although in this area the sensor returns very little information about the edges, the algorithm is still able to get good estimates.

Figure 9 shows a zoom-in of the trajectory center area. In this figure, the actual road's edges are also found within the uncertainty area. The area near the bottom part of the trajectory presents a larger error. This is due to the large amount of returns coming from trees that introduce extra uncertainty in the road estimation process.

Figure 10 shows the width of the road along the complete trajectory. This parameter varies from $11.5m$ to $28m$ corresponding the first of these values to the trajectory around the roundabout as shown in Figure 8. The width starts increasing when the vehicle is being driven through the wide curve shown in Figure 9.

Finally, Figure 11 displays the distance between the vehicle position and the left edge of the road. The small values are presented when the vehicle is turning to the left at the first curve and after passing the roundabout in Figure 8.

VIII. Conclusions and Future Work

This paper presented a probabilistic approach for road extraction using radar data. Radar information is modelled

Fig. 8. Estimated edges of the road describing the first and second curve located in the top left corner in Figure 7 along with the uncertainty area represented by cyan dashed lines. This area represents $\pm 2\sigma$ of the estimated roads.

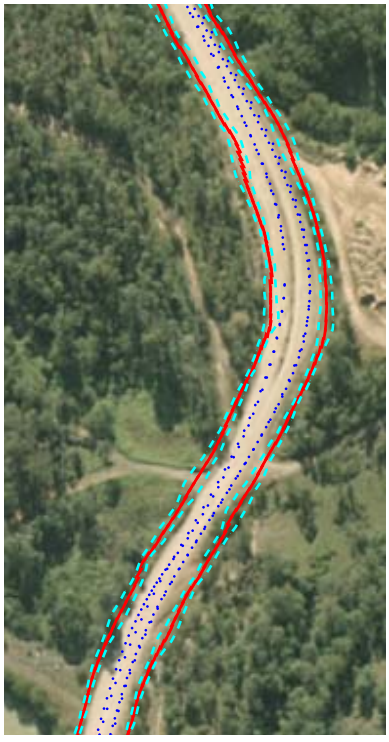

Fig. 9. Estimated edges of the road describing the third curve found in Figure 7 along with the uncertainty area represented by cyan dashed lines. This area represents $\pm 2\sigma$ of the estimated roads.

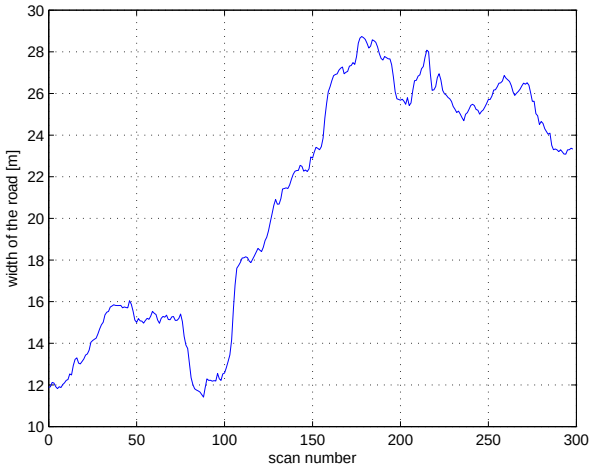

Fig. 10. Road width along the driven trajectory. The width varies between $12m$ and $28m$. These two values correspond to the width estimated along the roundabout in Figure 8 and the curve in Figure 9.

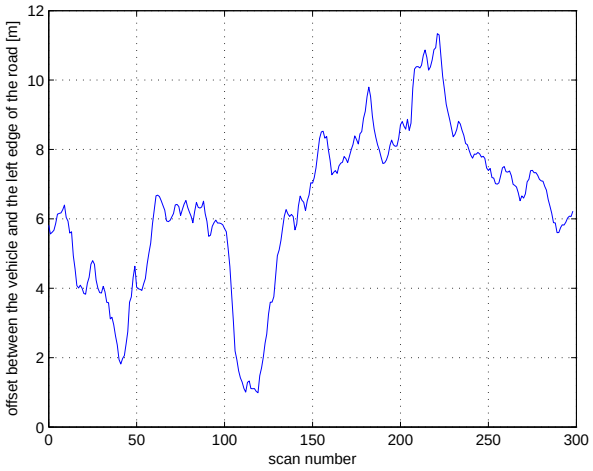

Fig. 11. Distance between the vehicle and the left edge of the road. The small values occur when the vehicle is turning to the left and passing the roundabout.

as Gaussian Mixture Models (GMM) where the uncertainty associated to each radar return was considered. A polynomial cubic approximation of a Clothoid function was applied to model the shape of the road. These clothoid functions were evaluated in the GMM, so that it was possible to compute the total cost for each proposed road.

Experimental results with data acquire in a mine environment validated the approach. A qualitative analysis of the results was done by superimposing the filter output on a satellite image. The good results obtained are very encouraging considering that no special work (such as adding infrastructure) was performed on the berms limiting the road. This implies that results could be improved by proper maintenance of the road. A system like the one presented will be essential for operator aiding under difficult environmental conditions or autonomous operations.

Future work will include the fusion of radar data with vision. The rich information provided by a visual sensor,

such as texture and colour will facilitate road discrimination, principally in clutter areas where the radar data is difficult to interpret. Furthermore, by using visual sensors such as stereo cameras, it is possible to recover the 3D structure of the environment, thus, an interpretation of the terrain could be obtained.

ACKNOWLEDGMENTS

We would like to thank to Graham Brooker and Javier Martinez from the Australian Centre for Field Robotics for providing the radar data and their help and support with the data processing.

This work has been partially supported by CONACYT and SEP Mexico, the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

REFERENCES

[1] S. Thrun and et.al, "Stanley: The robot that won the darpa grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[2] P. Beeson, J. O'Quin, B. Gillan, T. Nimmagadda, M. Ristroph, D. Li, and P. Stone, "Multiagent interactions in urban driving," *Journal of Physical Agents*, vol. 2, no. 1, 2008. [Online]. Available: http://www.jopha.net/index.php/jopha/article/view/14/13

[3] R. Risack, P. Klausmann, and W. Kruger, "Robust lane recognition embedded in a real-time driver assistance system," in *IEEE International Conference on Intelligent Vehicles*, Jan 1998, pp. 35–40.

[4] E. Nebot, J. Guivant, and S. Worrall, "Haul truck alignment monitoring and operator warning system." *"Journal of Field Robotics"*, vol. 23, pp. 141–161, 2006.

[5] K. Kluge, "Extracting road curvature and orientation from image edge points without perceptual grouping into features," in *Proceedings of Intelligent Vehicles Symposium*, 1994, pp. 109–114.

[6] K. Kluge and C. Thorpe, "The yarf system for vision-based road following." *"Mathematical and Computer Modelling"*, vol. 22, pp. 213–233, Aug 1995.

[7] R. Aufrere, R. Chapuis, and F. Chausse, "A fast and robust vision based road following algorithm," in *Proceedings of Intelligent Vehicles Symposium*, 2000, pp. 192–197.

[8] W. Wijesoma, K. Kodagoda, and A. P. Balasuriya, "Road boundary detection and tracking using ladar." *IEEE Transaction on Robotics and Automation*, vol. 20, pp. 456–464, Jun 2004.

[9] R. Aufrere, C. Mertz, and C. Thorpe, "Multiple sensor fusion for detecting location of curbs, walls, and barriers," in *Proceedings of Intelligent Vehicles Symposium*, 2003, pp. 126–131.

[10] B. Ma, S. Lakshmanan, and A. O. Hero, "Simultaneous detection of lane and pavement boundaries using model-based multisensor fusion." *IEEE Transaction on Intelligent Transportation Systems*, vol. 1, pp. 135–147, 2000.

[11] R. Lamm, B. Psarianos, and T. Mailaender, *Highway design and traffic safety engineering handbook*. McGraw-Hill, 1999.

[12] B. Southall and C. J. Taylor, "Stochastic road shape estimation," in *ICCV Eighth International Conference on Computer Vision*, 2001.

[13] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, 2004.

[14] N. J. Gordon, D. J. Salmond, and A. F. M. Simth, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE Proceedings-F*, vol. 140, no. 2, pp. 107–113, Apr. 1993.

[15] M. Isard and A. Blake, "Icondensation:unifying low-level and high-level tracking in a stochastic framework," in *ECCV European Conference of Computer Vision*, 1998.

[16] G. Kitagawa, "Monte carlo filter and smoother for non-gaussian non-linear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. 1–25, 1996.

[17] G. Brooker, "Understanding millimetre wave fmcw radars," in *International Conference on Sensing Technology*, 2005.