Distributed motion planning for ground objects using a network of robotic ceiling cameras*

Andreagiovanni Reina, Gianni Di Caro, Frederick Ducatelle, Luca Gambardella

Dalle Molle Institute for Artificial Intelligence (IDSIA)
Galleria 2, 6928 Manno - Lugano, Switzerland
{gioreina, gianni, frederick, luca}@idsia.ch

Abstract. We study a distributed approach to path planning. We focus on holonomic kinematic motion in cluttered 2D areas. The problem consists in defining the precise sequence of roto-translations of a rigid object of arbitrary shape that has to be transported from an initial to a final location through a large, cluttered environment. Our planning system is implemented as a swarm of flying robots that are initially deployed in the environment and take static positions at the ceiling. Each robot is equipped with a camera and only sees a portion of the area below. Each robot acts as a local planner: it calculates the part of the path relative to the area it sees, and exchanges information with its neighbors through a wireless connection. This way, the robot swarm realizes a cooperative distributed calculation of the path. The path is communicated to ground robots, which move the object. We introduce a number of strategies to improve the system's performance in terms of scalability, resource efficiency, and robustness to alignment errors in the robot camera network. We report extensive simulation results that show the validity of our approach, considering a variety of object shapes and environments.

1 Introduction

Path planning is a core problem in robotics (see [9] for an overview). In its basic version, the path planning problem consists in the definition of the optimal sequence of rotations and translations needed to move an object of a given geometry from an initial to a target configuration while avoiding collisions with obstacles. If the constraints on the motion only depend on the environment's obstacles and on the relative position of the moving object, the problem is *holonomic*. In *non-holonomic* motion planning also dynamic constraints are considered.

In this work, we focus on holonomic path planning in the following setting. An object of assigned shape has to be moved from an initial to a final location in a large cluttered area. A high movement accuracy is required, up to a few centimeters precision, in order to effectively avoid the existing obstacles. No a priori knowledge about the environment is available. In order to acquire and

* This research was partially supported by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics

process the information required for planning the motion of the object on the ground, we propose the use of a *network of cooperating cameras* with a *top view* of the area where the object can be moved. We assume the camera network to be able to autonomously take position in the environment. For this purpose, as reference model for the camera nodes, we used a *swarm of flying robots*. The swarm can be deployed in formation such as to cover through the vision system the entire area between the initial and final positions, and to be able to locally communicate. Then, they attach to the ceiling and keep stationary positions for the entire process. More technical details are given in Section 2. This architecture is suitable for path planning problems in *large areas*, where a single camera is not sufficient to effectively cover the entire area, and a ceiling camera network can be effectively deployed. Examples are factories, warehouses, or malls: large indoor areas, characterized by the presence of relatively narrow alleys and turns, and irregularly spread, and often dynamic, obstacles.

In our *distributed architecture*, each camera node plays the role of a *local planner*: it plans the motion relative to the area that it sees, and locally exchanges information with its neighbor nodes through a wireless channel in order to merge and organize the local views into a global feasible plan for the object on the ground. Compared to a single camera solution, this approach determines *sensing errors*, due to intrinsic uncertainties in the calculation of the relative positions of the cameras and the overlapping in their local views, and *efficiency issues*, due to communication and coordination overhead. In this work, implement a fully distributed system for effective path planning building on and extending established approaches for path planning. Then, we consider error and efficiency issues. We cope with the latter by introducing a number of heuristic strategies. On the other hand, since sensing errors are an intrinsic property of the system, we consider them as internal parameters, and we evaluate how they influence the performance. The aim of this work is to show that our distributed planning system can find near optimal paths, makes an efficient utilization of computational and communication resources, is robust to increasing sensing errors, and its performance scales with the size of the environment and the number of cameras. We report extensive simulation results that precisely show these properties considering objects of different shapes and a variety of cluttered environments.

The article is structured as follows. In Section 2 we define the scenario characteristics, the initial assumptions and the camera network model. Section 3 discusses related work. Section 4 explains the planner architecture, describing the different phases of the process and the optimization heuristics we propose. Section 5 shows experimental results, and section 6 concludes the paper.

2 Scenario characteristics and camera network model

We use a *swarm of flying robots* to implement the camera network used for distributed planning. The robots are modeled after the *eye-bot* [10], an indoor flying robot developed in the EU-funded project *Swarmanoid* (`http://www.swarmanoid.org`). The eye-bot can passively attach to the ceiling using a magnet (the design assumes the presence of a ferrous ceiling). It has a pan-and-tilt camera, which can be pointed in any direction below it. It can communicate using both wi-fi and a line-of-sight infrared system [10], which also provides it with relative positional information about other eye-bots (distance and angle).

The mobility and autonomy of the robots can be exploited to deploy the camera network over the area where the object moves. The robots can select their stationary positions at the ceiling to let the swarm formation effectively cover the area with the combined camera views, while, at the same time, observing that neighbor robots are in wireless communication range. We do not study how the formation can be obtained; any algorithm that lets robots spread in an area or find a target location from a given source location (e.g., [11]) could be used.

Using the pan-and-tilt unit, the robots orient their camera to look at the area directly below them. Each robot computes a 2D occupancy map for the obstacles below it, and the path of the ground object is computed with respect to this 2D projection.[1] The field of view of each robot must overlap with that of its neighbors. The size of the overlapping area must be greater or equal to the dimension of the moving object. This is required in order to connect locally calculated sub-paths (see Section 4). For the same reason, knowledge of the relative position and orientation between neighbor cameras is also required. The eye-bots can compute this information autonomously, using the range and bearing capabilities of the infrared system.[2] This means that they can also adapt this information when the eye-bot network topology changes. Clearly, the relative distance and angle information acquired through the infrared system are affected by errors (as would be the case with any other relative positioning system). In our simulation model of the eye-bot, we model distance and angle error values using two different zero mean Gaussian distributions with configurable variance values. In this way, we model error estimates in rather general and, at the same time, realistic terms, while avoiding to be too hardware-specific. This approach allowed us to study the impact of these error estimates on distributed path planning performance for a wide range of error values (see Section 5).

Finally, we point out that our focus in this paper is only on the distributed calculation of a path plan. We assume that the start and target locations are given as input, and the object can be moved by a system (e.g., a robot) that can interpret the path planning instructions and can check its correct actuation.

3 Related work

The path planning problem has been extensively studied considering various formulations and solutions. References and discussions can be found in [8, 3, 9]. Our distributed planner is based on classical work in path planning [8, 1]: it is derived from the numerical potential field technique for a single camera planner. The potential field is computed using the *wave front expansion with skeleton*

[1] In this way, we classify the areas covered by table-like obstacles as fully inaccessible, while the object could pass under, depending on its volumetric dimensions.

[2] We assume that all cameras have the same calibration and size of field of view.

on a uniform cell partitioning of the 2D map of the environment. This solution first spreads the potential over a subset of the free space, called *skeleton*, which corresponds to the *Voronoi diagram* [3, 5, 12]; then, the potential is computed in the rest of the map. The potential descent from the start to the final configuration is performed using A^*[6]. The moving object can have any shape and dimension. We follow the approach proposed in [3, 7], modeling the object with a discrete set of *control points*. Various ways to apply A^* to the control points exist; we sum the potential over the different points to compute the distance estimation.

Centralized path calculation is the most common approach to planning, even in the presence of multiple cameras. All partial maps are sent to a central node and are fused to construct a global map. Several methods have been proposed for *map reconstruction* based on feature correlation in partial maps [2]. Compared to these methods, our distributed collaborative approach limits the use of communication and computational resources, and relies on autonomous relative positioning, resulting in more robust, efficient and scalable behavior.

Our system can rapidly build the map of the environment thanks to the camera coverage of the area. Moreover, it has the potential to rapidly react to changes in the environment, and to provide precise localization inside it. In this respect, it is equivalent to a *SLAM* [4] process. In fact, as a possible alternative, a swarm of ground robots could be used for SLAM, and build a free space map of the environment that could be more accurate than the one built by our camera network (see Footnote 1). However, the use of ground robots to build accurate maps of large cluttered environments requires the commitment of extensive resources, for time and computation. Our solution provides better efficiency, but we pay this advantage in terms of a potentially lower map accuracy.

4 Distributed path planning

The distributed path planning algorithm that we propose is an extension of the centralized algorithm described in Section 3. The algorithm consists of 3 phases: (i) *neighbor mapping*, (ii) *potential field calculation*, (iii) *path calculation*.

Neighbor mapping. Using the range and bearing system, each node builds a *neighbor table*, in which the relative positions and orientations of neighbor nodes are stored. Assuming that all cameras have a field of view of the same size, each node n uses this information to project the field of view of each neighbor m on its own reference system. This way, n builds an estimate of the *overlapping region* between its own field of view and that of m. In particular, n first segments its local map into a discrete set of cells, and then identifies which cells lay on the edges of the field of view of m, and which cells of its edges lay on m's field of view. Hereafter we refer to the former cells as *shared edges* and to the latter as *open edges*. Figure 1 illustrates this process.

Potential field calculation. The second phase is the calculation of a potential field. It is based on a diffusion process that starts from the destination point and iterates between the nodes until the potential is diffused on the entire map. This process is implemented in the following distributed way.

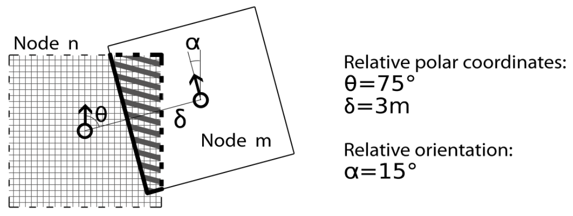

Fig. 1. Estimating θ, δ and α, n builds a projection of m's field of view to define: overlapping (striped region) *shared edges* (solid bold lines), *open edges* (dashed lines).

Each node first calculates the local *skeleton*, which corresponds to the *Voronoi diagram* on the 2D occupancy map (i.e., the *maximum-clearance roadmap* for object motion [9]). [3] Once the local skeleton is defined, each node calculates the *local potential field* through a cooperative diffusion process. The potential field defines a force that attracts the object towards the destination, and at the same time repels the object away from obstacles. In our algorithm, the potential is a scalar function that defines the attraction/repulsion intensity for each cell. The function has a global minimum in the destination point, and maximum value on the obstacles. In all other cells the function decreases towards the destination, such that the planner can aim to the goal following the gradient descent direction. The nodes that see the destination begin potential diffusion by assigning a zero value to the destination cell, and then incrementing the potential by one per cell. At first, nodes diffuse the potential only on the set of the skeleton cells; this way, the value assigned to each cell corresponds to the distance from the current cell to the destination over the skeleton. Then, the potential is calculated on the remaining free cells: diffusion starts from the skeleton, and values are increased while moving away from it and getting closer to obstacles.

Once the potential field is completely spread over the local map, each node n sends to its neighbors the values of the shared edges cells that they have in common. To minimize communication overhead, only shared edges cells that belong to the skeleton are transmitted. This way, each neighbor m of n receives a set of n's skeleton cells. Since n has an estimate of the relative position and orientation of m, the coordinates of these cells are expressed relative to m's frame of reference. m replaces the value of the corresponding cells in its own local map, and connects these new skeleton cells to the local existing skeleton. Then, m uses the received skeleton cells as starting diffusion points for spreading and updating the potential field on its local map.

The nodes that can see the destination maintain P potential field maps, one for each of the P control points of the object. This way, these nodes can

[3] The environment's skeleton resulting from all local skeletons differs from that which would be calculated in a centralized way using a global map. The differences are due to the fact that, during skeleton calculation, the frontiers of a (local) map need to be considered as obstacles. Therefore, at the corners of each map the local skeleton shows bifurcations that are not present in the centralized skeleton.

correctly position the object to the expected final configuration. The other nodes keep a single potential field, relative to the control point positioned in the object center. This means that these nodes do not take into account the object final orientation. This strategy optimizes the use of computational resources without affecting in practice the quality of the solution.

Path calculation. Once the potential field is spread among all nodes, the node above the start position starts path calculation: it plans the partial path relative to its local map using the values of the potential field. The A^* algorithm is used to identify a gradient descent trajectory in this field. A^* searches the solution by building an *exploration tree* of configurations. Each configuration defines a position and orientation of the object, and has a cost value equal to the sum of the distance from the initial point to the current configuration (measured in number of crossed cells) and the estimated distance to the destination (calculated as the sum of the potential value of the cells occupied by the control points). At each step, the algorithm selects the leaf of the tree that corresponds to the configuration c with the lowest estimated cost. Then, it calculates all the configurations that can be obtained from c by a one-step movement (a rotation or translation)[4] and their estimated costs, and adds them to the tree as leafs of c. The process iterates until the goal configuration is visited (*success*), or the exploration tree is completely expanded (*failure*), or the selected configuration falls outside the area of view laying on an open edge. In the latter case, the node sends the object coordinates to the neighbor m with which it shares the open edge. m continues the process, using as start position the received configuration. Passing object configuration from one node to another is made possible by the overlapping between neighbors' maps. However, this overlapping is subject to errors deriving from camera relative position estimation errors (see Section 2). These errors can affect the system performance, as we study in Section 5.

Since potential diffusion does not explicitly consider the dimensions of the moving object, it is possible to get trapped in local minima during path calculation. This is a well-known problem: following the potential might cause the object to get stuck in an area that is too narrow to let it pass. Our basic algorithm deals with the issue by locally backtracking and trying out alternatives. If a node fails in local planning, it sends to the preceding neighbor in the planning a *local failure* message. At the receiver, this message triggers the resuming of the local tree exploration process to calculate an alternative path. This approach is expensive, both in computation and communications. In order to improve the efficiency of this process and minimize the probability to get trapped in local minima we propose two heuristics (see Section 4.2).

Each node involved in the path calculation process stores: (i) the previous neighbor (from which the node received the entrance path), (ii) the planned local path, and (iii) the next neighbor (to which the node transmitted the object configuration of its local path). The distributed algorithm executes an exhaustive search of the solution: if the process ends with failure it means it has visited all

[4] In our system, a one-step translation corresponds to one cell, and a one-step rotation corresponds to an angle α. In the experiments, cell size is 10×10 cm, and $\alpha = 15^o$.

possible configurations that are reachable from the starting point but no feasible solution exists given the characteristics of the calculated potential field and the selected search parameters (e.g., cell discretization, minimal rotation angle).

4.1 Adaption to changes in the environment

An important advantage of our distributed system is that it can locally and quickly detect and adapt to a change in the environment (e.g., a change in obstacles' position, or the appearing/disappearing of an obstacle). A centralized system would correct the Voronoi skeleton, repeat the potential field diffusion and restart the path planning. In our distributed architecture, the system can reduce the re-initialization costs by replanning only a limited part of the path. A node that detects a change in its local map informs the other nodes only if an alternative local partial path cannot be found.

More specifically, if the change happens when the system has completed the path calculation and the navigation has already started, the desired behavior is a fast response to adapt the path and allow the successful completion of the navigation task. A node n that detects a change in its local map, first tries to locally plan an alternative path by generating a new local potential field, with the constraint of maintaining fixed the original entrance and exit configurations. If a feasible path cannot be found, n notifies the destination node (through multi-hop wireless communication) to trigger a new potential field diffusion process. The destination node decides either to repair the path (from n to the end), or to recalculate the entire path (using the current position of the navigating object as new starting position). The decision for either alternative is taken in relation to the position of n in the sequence of nodes along the path. E.g., a local repair is issued when n is close to the final

4.2 Heuristics to reduce local minima attraction

The *skeleton pruning* and *local minimum detection* heuristics aim to improve the efficiency of the system minimizing the probability to get stuck in local minima.

Skeleton pruning. We prune the skeleton during potential field diffusion to block passages narrower than a predefined width w_{sp}. We set w_{sp} to the width of the smallest dimension of the moving object. Since size is not the only parameter defining whether an object can cross a passage (e.g., it depends also on the object's morphology) or not, the heuristic does not guarantee to remove all local minima. Setting w_{sp} higher, however, we may remove feasible paths.

Local minimum detection. The heuristic is executed during path calculation, when a node detects a local minimum due to the presence of a too narrow passage. First, the node places a virtual obstacle over the cells of the passage. Then, it triggers a new distributed potential field diffusion step. The assumption behind the heuristic is the following: the final solution obtained by recalculating the whole potential field and restarting the distributed path planning with the new information is expected to be of better quality than that obtained by exhaustively searching a way to escape from the local minimum.

4.3 Heuristics to optimize the use of resources (efficiency heuristics)

Even using the above heuristics, dealing with *local minima* causes a high computation and communication load to the system. Moreover, given that the overall path is computed by combining local paths without using any global path information, it can contain *loops*. We propose two *efficiency heuristics* to deal with these issues: *block cells* and *smart loop avoidance*.

Block cells. During path calculation, if a node n has completely expanded its exploration tree in its local area without finding a valid path, it sends a *local failure* message to the preceding neighbor node m in the path. m resumes the path planning process and tries out another exit configuration, which may be in the same direction of attraction toward n. This way, a repeated exchange between n and m can take place until all exit configurations towards n are tried out unsuccessfully. To reduce communication and processing overhead associated to these situations, after a local path failure, m identifies the entry cells that have generated the failure in neighbor n, and avoids to send the object again through the same cells by removing them from the available open edges.

Smart loop avoidance. Given that $(n_1, n_2, \ldots, n_k)$ is the sequence of nodes associated to the computed path, if $n = n_i = n_j$, for any $1 \leq i, j \leq k,\ i \neq j$, then a loop is said to be present in the path if the two configurations to enter n at steps i and j can be connected together within n's local area. Therefore, the sub-path between n_i and n_j can be conveniently removed. This is done in the following way. Controlling its internal path components a node n checks whether a loop is present and can be removed. In this case, the node n sends a *loop* message to its previous node m in the path. After receiving the loop message, m deletes its local path and forwards the loop message to its preceding node. The process is iterated until the loop message reaches again node n and the loop is completely removed from the path. In order to avoid the re-creation of the loop, n perturbs its local potential field, and then resumes the path planning process.

5 Experimental results

We test our distributed algorithm in simulation considering a large set of sample scenarios with varying area dimensions, position and number of eye-bots, shapes of moving object, and obstacle positions (Figure 2 shows two examples). We studied the performance of the proposed planning solutions in terms of *effectiveness, efficiency, scalability,* and *robustness to alignment errors*. As performance metrics we selected *success ratio*, the percentage of successful runs, and *path quality*, the relative length of the path compared to the path calculated by a *centralized algorithm with complete and perfect knowledge*. We study how performance varies in function of the alignment error between the cameras (angle and distance errors). Errors are modeled using a zero mean Gaussian distribution (in the experiments described in the following, we report the error as the standard deviation of the corresponding distribution).

In each scenario the robots are deployed in a grid formation that covers the area. The distance between robots is 2 m. The visual field of each robot is the

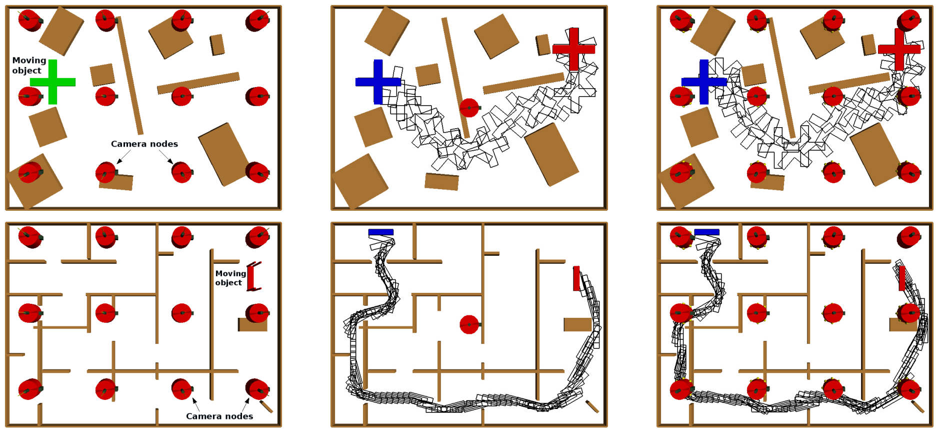

Fig. 2. (left) Examples of the scenarios used for experiments. (center) Reference paths from a global planner with perfect knowledge. (right) Paths of the distributed planner

4×4 m^2 area below their camera. Each point in the data plots is the average of 40 runs over 25 different scenarios including different numbers of robots (ranging from 12 to 36). The spread of the values around the average (not shown) is quite large, due to the strong heterogeneity of the scenarios. However, we assessed the significance of the results and the differences among the different versions of the algorithm through statistical testing. Each scenario is characterized by a different random positioning and structure of the obstacles. The percentage of the area occluded by obstacles vs. the total is approximately 18%. The shape of the moving object is randomly selected among rectangular, cross, and L shapes.

5.1 Robustness to alignment errors

The effect of camera alignment errors is shown in Figures 3 (angle error) and 4 (distance error). They report the performance of different versions of the algorithm: without heuristics, only with efficiency heuristics, and with all heuristics. The results show that for relatively low errors the performance is always very close to the reference. For increasing errors, the performance of the algorithm with heuristics degrades rapidly for success rate but slowly for path quality. Without heuristics the behavior is opposite. In both cases, the system is relatively sensitive to errors on the angle, while it is quite robust to distance errors.

In the experiments where the error is zero, the paths of the system without heuristics are on average 25% longer of the reference path. This is partially due to the generation of loops (which can be also caused by the presence of local minima). The use of heuristics mitigates these problems and reduces path length. However, the skeleton difference with respect to a global path planner (see Footnote 3) cannot be avoided, resulting in unnecessary movements for the object when it is moved between nodes, and, therefore, in slightly longer paths.

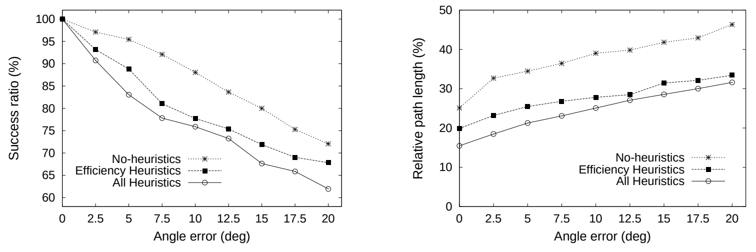

Fig. 3. Performance evaluation vs. increasing error in angle estimation.

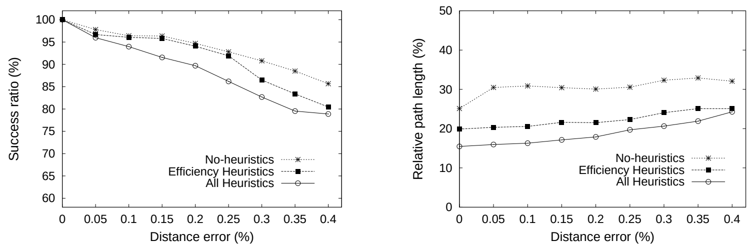

Fig. 4. Performance evaluation vs. increasing percentage error in distance estimation.

5.2 Communication efficiency

Communication efficiency is a fundamental aspect of the system because it impacts scalability. We quantify the use of communication resources through the average number of messages that each robot sends during the path planning process. In these experiments, the robots maintain a 4×3 grid formation.

The plots of Figure 5 show the results for this set of experiments. As expected, the system with *efficiency heuristics* has the best efficiency. It is able to maintain low values of exchanged messages even with the increase of the alignment error. The system with all the heuristics makes more intensive use of communication due to the strategies applied for local minima avoidance.

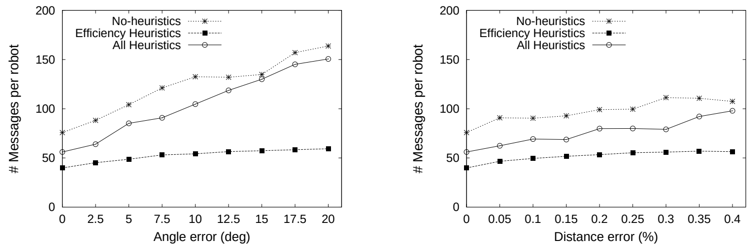

Fig. 5. Use of communication resources vs. error in angle and distance estimation.

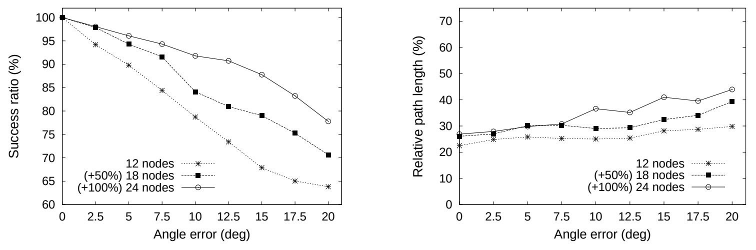

Fig. 6. Planner with heuristics: effect of node density vs. error in angle estimation.

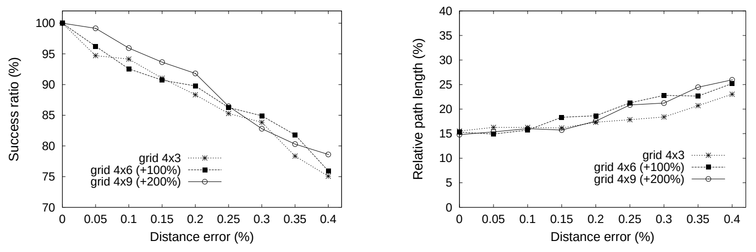

Fig. 7. Planner with heuristics: effect of environment size vs. error in angle estimation.

5.3 Scalability: increasing robot density in a fixed size environment

We study the performance of the distributed system with respect to an increase of system's resources: in Figure 6 we show the effect of increasing the density of the nodes over a fixed area while varying angle errors. Increasing node redundancy allows the system to deliver a higher success ratio (Figure 6, left). This effect is more marked in the experiments with larger errors, in which the presence of a larger number of robots can balance the effect of these errors to find alternative valid paths. On the other hand, the increase in the density leads to longer paths, as shown in results of Figure 6 (right).

The results for the performance of the algorithm without the heuristics (not reported) show a similar behavior, with both the graphs being shifted up (higher success ratio and longer paths). The results for the error in distance (not reported) are analogous to those of Figure 6, but show a better robustness of the system to the error, as already observed in the previous experiments.

5.4 Scalability: increasing environment size, fixed density of robots

We studied the scalability of the system increasing environment size and maintaining robot density constant. The area of the basic scenario is 10×8 m^2, with a coverage of 12 robots in a grid formation 4×3. We use this basic scenario and two additional ones in which the area is scaled by a factor 2 and 3 respectively. The plots of Figure 7 show the results: the trends and values in the three scenarios are very similar. This means that the system is able to deal with an increase of the environment size with an increase of resources (number of robots) without degrading performances, showing its scalability.

6 Conclusions and future work

We have proposed a novel system for distributed path planning, which calculates with high accuracy the sequence of roto-translations of rigid objects of any shape moving through cluttered areas. The system architecture uses a swarm of flying robots, which deploy in the environment and form a distributed camera network. The robots solve the path planning problem cooperatively, through local calculations and wireless message exchanges. We adapted well-known solutions for path planning to our distributed architecture. Compared to systems with single cameras or centralized computations, the fully distributed approach can be more scalable, flexible, and robust. However it introduces efficiency issues and sensory errors. A number of heuristics were proposed to enhance the system's efficiency and effectiveness. In a wide range of simulation experiments, we show that the system is efficient, scalable, and robust to alignment errors between cameras.

Future work will include in the first place the implementation and testing of the system on real robots. We will also extend it and improve it, and perform tests with dynamic obstacles (e.g., including humans).

References

1. J. Barraquand, B. Langlois, and J. Latombe. Numerical potential field techniques for robot path planning. *IEEE Trans. on Sys., Man and Cyb.*, 22(2):224–241, 1992.
2. A. Birk and S. Carpin. Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE, special issue on Multi-Robot Systems*, 94(7):1384–1397, 2006.
3. H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, Cambridge, MA, 2005.
4. H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): The essential algorithms. *IEEE Robotics and Automation Magazine*, June 2006.
5. S. Garrido, L. Moreno, and D. Blanco. Voronoi diagram and fast marching applied to path planning. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006.
6. P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Sys., Sc. and Cyb.*, 4(2):100–107, 1968.
7. J.-C. Latombe. A fast path planner for a car-like indoor mobile robot. In *Proceedings of the 9th National Conf. on Artificial Intelligence*, pages 659–665, 1991.
8. J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic, 1991.
9. S. M. LaValle. *Planning Algorithms.* Cambridge University Press, 2006.
10. J. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano. 3-D range and bearing sensor for collective flying robots. *Journal of Field Robotics*, 2011. (Submitted).
11. T. Stirling, S. Wischmann, and D. Floreano. Energy-efficient indoor search by swarms of simulated flying robots without global information. *Swarm Intelligence*, 4(2):117–143, 2010.
12. O. Takahashi and R.J. Shilling. Motion planning in a plane using generalized voronoi diagrams. *IEEE Trans. on Robotics and Automation*, 5(2):143–150, 1989.