A Distributed Cognitive Map for Spatial Navigation Based on Graphically Organized Place Agents

Jörg Conradt
Institute of Neuroinformatics, UZH/ETH-Zürich

Abstract

Animals quickly acquire spatial knowledge and are thereby able to navigate over very large regions. These natural methods dramatically outperform current algorithms for robotic navigation, which are either limited to small regions [1] or require huge computational resources to maintain a globally consistent spatial map [2]. We have now developed a novel system for mobile robotic navigation that like its biological counterpart decomposes explored space into a distributed graphical network of behaviorally significant places, each represented by an independent "place agent" (PA) that actively maintains the spatial and behavioral knowledge relevant for navigation in that place. Each PA operates only on its limited local information and communicates only with its directly connected graphical neighbors. Thus, there is no global supervisor and it is only necessary to maintain spatial consistency locally within the graph. This simple strategy significantly reduces computational complexity; scales well with the size of the navigable region; and permits a robot to autonomously explore, learn, and navigate large unknown office environments in real time.

Introduction

Navigation is easy - we do it every day. We can effortlessly explore areas we have never been before, memorize important places and return to those again and again. But how does this work? How do we build up, represent, and use information about space? This thesis presents a biologically plausible principle for acquiring, storing, maintaining and ultimately using spatial knowledge for short and long-range navigation between behaviorally significant places: a *cognitive map* [3].

Navigation in Engineering and Neuroscience

In today's engineered navigation systems [2], an active agent such as a robot typically stores all acquired information about its spatial environment in a global map relative to an absolute origin [4, 5], shown in Figure 1, left. Adding or updating information and using stored knowledge for navigation typically require considerable computational resources and involve a single active agent - such as a computer program - having access to all accumulated information. Alternatively, several topological approaches for navigation exist [6, 7], shown in Figure 1, middle, and also hybrid topologic-metric approaches [8-10]. But all these approaches rely on a single active agent that is reasoning based on all previously acquired data. Such a mechanism is unlikely to be performed in animal or human brains: there does not appear to be any one active region of our brain "inspecting" other passive regions to use stored information for navigation.

In the last few decades place cells and - much more recently - grid cells have been in the center of neuroscientific research about navigation [11, 12] and its robotic implementation [1, 13, 14]. These families of cells in Hippocampus [11] and Entorhinal Cortex [15] show significantly increased activity whenever an animal happens to be within a well defined region in an experimental setup. It is widely agreed that activity of a collection of such place cells represents the animal's current spatial position within a local frame of reference. These cells show a stable firing pattern over a long time within an environment, perform a complete remapping of their firing pattern when the observed animal enters a distinct new environment[16], and revert to the previous firing pattern upon returning to a familiar environment. However, experiments so far are constrained to relatively small areas of about 2m in diameter, whereas wild animals typically live in areas of several thousands of square meters. It is yet to be investigated how navigation in larger areas impact firing patterns in place cells.

A Distributed Cognitive Map

In this thesis we explore a new principle of perceiving, maintaining and operating on spatial knowledge: every element of the system operates exclusively on locally available information and is constrained to decide actions based only on knowledge about its vicinity. We do not represent space in a global consistent data structure as traditional topological or metric-based approaches do. Instead, starting without prior spatial knowledge, our system autonomously creates a graphical network of independent "patches of knowledge" at behaviorally relevant places. Such patches contain spatial and behavioral information only about their local environment. They actively control the physical agent - a robot - in their vicinity, and maintain and update their limited knowledge about their environment, instead of serving as passive data storage containers. As these active patches know about and communicate with their local nearest neighbors, they implicitly establish the topology of a network that represents behaviorally significant space. Each of these nodes of the network is unaware of its position within the network and the position it represents in global space. No process in our system maintains or operates globally on the network; in fact, the whole network only exists because of message-passing between autonomously acting neighboring nodes. This principle is illustrated in Figure 1, right.

Note that the distributed system does not explicitly represent cycles in the environment. Figure 1, left and middle, show a cycle in the environment between places D-E-G-D, which is computationally expensive to handle [2]. In the distributed representation, no actor has access to D, E, and G simultaneously. The system is unaware of the cycle, but still knows all individual traversable paths that together constitute the cycle.

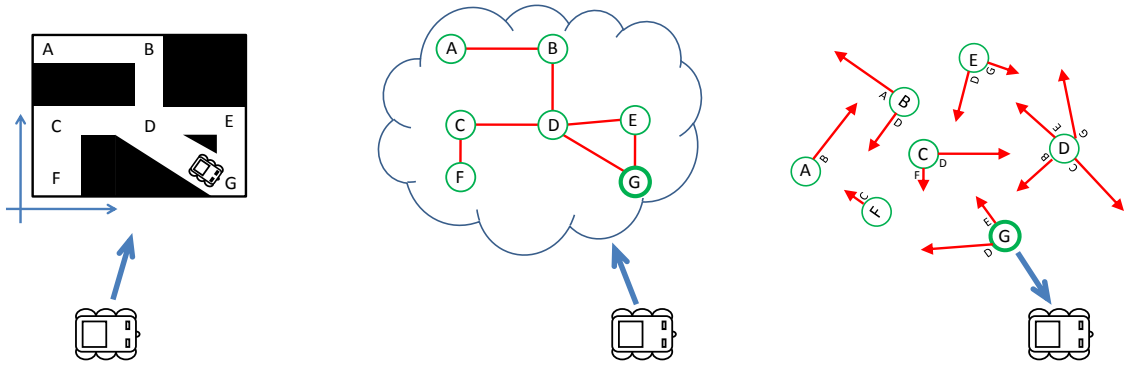

Figure 1: left: a global Cartesian map showing a top-down view of an environment, maintained by a single computer or a mobile robot. Middle: topological representation of the same environment recorded as a consistent data structure. A computer program/robot maintains this structure and operates based on stored information. Right: A collection of individual programs as active place-representations, each only knowing their respective local space and direct neighbors. The true graphical structure exists only implicitly in the collection of all places, to which no actor has access. At any time only a single one of these individual programs interacts with the robot; i.e. it directs the robot and obtains sensed information about the current local environment.

Implementation and Results

The system we developed starts without spatial knowledge in an unfamiliar environment which it automatically explores using a mobile robot. While exploring, the robot constantly maintains a representation of its current local environment[1] which consists of fused information from various on-board sensors. When detecting a behaviorally relevant stimulus[2], a new place agent captures a snapshot of the current local environment, records the previously active place agent as a neighbor, and directs the robot to further explore unknown regions. Repeating this process incrementally builds up a collection of independent place agents. All these agents only know their local spatial environment and communicate with their direct neighbors, as shown in Figure 1, right. Despite these severe constraints on knowledge and communication, the collection of all individual place agents shows emergent globally consistent behavior without having a single globally acting entity in the system. An example of such global behavior is the guidance of the robot to a previously recorded target represented elsewhere in the network.

Our system has not only successfully explored spaces as small as a single room, but also our whole institute of about 60x23m (Figure 2). Only few operations require time scaling linearly with the number of nodes in the network; most operations are performed on local data only. Currently, all place agents required to represent our institute run in real-time on a single computer; but as they are independent of each other they could be distributed among multiple possibly less powerful computing units. We are convinced that we can map significantly larger environments without suffering degraded performance.

[1] "Current local environment" refers to space within a few times the robot's body-length, typically within sensor reach. The overall operating area of the robot, in contrast, covers several 100 times the robot's body-length.

[2] Such as an intersection of paths, a battery charger or fresh coffee, depending on one's preference.

We believe that neural hardware is well suited to implement such an “active distributed cognitive map”, as brains are intrinsically distributed processing systems without global access to all memorized information - which is required by traditional algorithms for navigation. We therefore believe that the proposed system resembles biological information processing much more closely than current methods for long-range spatial navigation.

Relating our system to place-field neurons found in Hippocampus [11, 12], we interpret observed stable activity-patterns of place-cells within a small experimental setup as an indicator for a short-range local map in animals brains. Such a local map is implemented by a single place-agent in our system. When an observed animal changes to a different place, a complete remapping of firing patterns across place-field neurons occurs [16], which corresponds to transferring behavioral control to a new place agent in our model.

Our system explains results from typical behavioral experiments with rats [3, 17], without requiring a global actor such as a computer program or an omni-conscious homunculus having access to all acquired information.

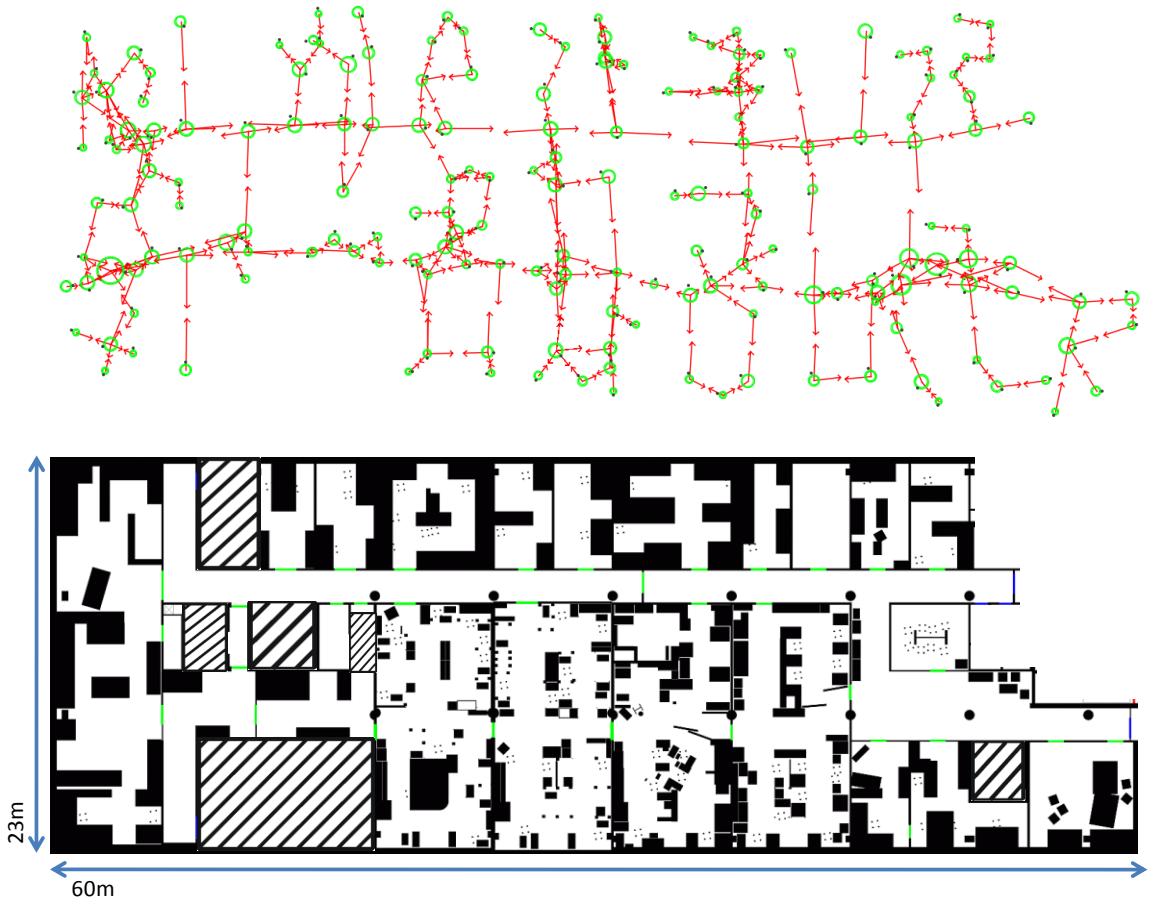

Figure 2: Top: A distributed graphical representation of our institute established by 177 independent place agents (PAs, green circles). The spatial arrangement of PAs only happens for this display; no actor in the system is aware of the structure. Red arrows show 45% of the recorded distance to a neighboring PA. Places of behavioral relevance are covered densely with nodes, whereas places such as long aisles show sparse coverage. Bottom: a plan-view of the institute (60x23meters).

This short summary continues with a more detailed description of individual conceptual blocks that compose our system:

- Fusing and maintaining sensor data for local place signatures
- Creating independent active place agents (PAs) that represent behaviorally relevant space and communicate with neighbors to establish a network of relevant places
- Performing behaviors with such a distributed graphical representation

Creating Place-Signatures to Represent Local Environmental Data

The system outlined above needs to represent spatial information at two different levels: While a mobile robot moves in its environment it permanently reports perceptions from various on-board sensors that accumulate in memory (Figure 3, left). The accumulated information changes quickly as the robot moves; it represents the robot's currently perceived local environment, typically limited in size to a few times the robot's body length. On a different level, place agents constituting the network described above need to remember information about their local environment which hardly changes over time, and extract behaviorally relevant information - such as free space - from previously acquired spatial knowledge (Figure 3, right).

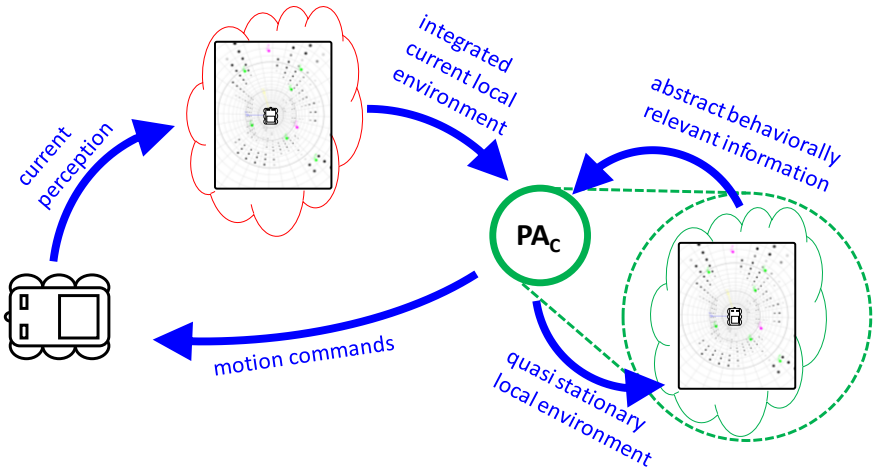

Figure 3: Representing spatial knowledge: A data representation "EnvInf" (red cloud) accumulates sensor perceptions from a mobile robot and provides an integrated representation of the current local environment to the place agent currently in control of the robot (PA$_C$). Each PA maintains information about its local environment in an internal instance of EnvInf (green cloud). A PA can request behaviorally relevant information from its instance of EnvInf - such as open space - to direct the mobile robot.

We have developed a data representation "Environmental Information" (EnvInf) that fulfils both these purposes. For the first, the mobile robot constantly adds perceived sensor information to a single instance of EnvInf, while EnvInf itself inspects and maintains all accumulated information. Internally, EnvInf creates individual layers to store information for each type of sensor, adding consecutive data from the same sensor to the same layer. Each layer is implemented as a binned log-polar occupancy grid, since for spatial information exact positions are more relevant close to the center than further away. Maintenance of all data, e.g. decaying and translating previous

information according to the robot's motion, is performed by EnvInf internally. Decaying old data reduces its significance and puts a fixed upper bound on memory requirements, while translating old data keeps it valid with respect to the current frame-of-reference when the robot moves. The robot behaves as a passive agent; it only delivers elementary sensor information to an instance of EnvInf, which upon request provides an integrated snapshot, called a place signature (PS) of the robot's current local environment (Figure 4).

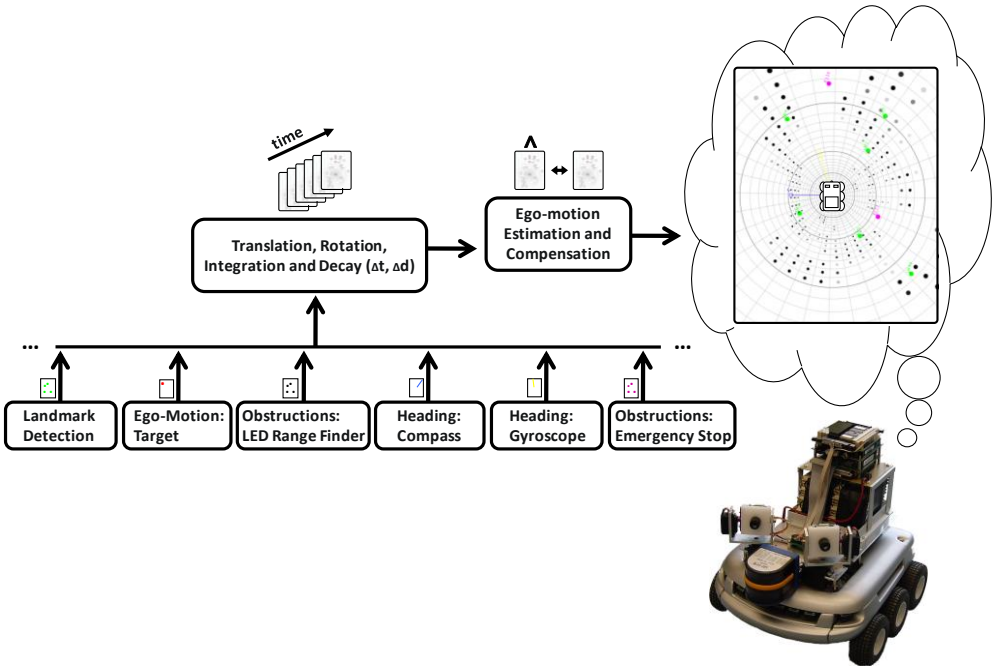

Figure 4: Sensor fusion from on-board-sensors (row of six black boxes). All information gets translated and rotated according to the robot's motion, decayed in time and distance, and integrated. The resulting short-range "place signature" (inside cloud) is the current best estimate of the robot's local environment in plan-view.

Additionally, EnvInf allows active place agents (PAs) to remember their quasi-stationary local environment: a new PA memorizes the current local place signature and stops the internal time decay. Future updates of stored information, when the robot revisits a place and provides a new place signature, keep information consistent with the local environment. Besides basic environmental knowledge, a PA can store and maintain abstract information in the same instance of EnvInf, such as directions and distances to neighbors in the network with respect to its own coordinate frame, or behaviorally relevant values such as the existence of a battery charger.

The PAs of the network do not interpret stored data. Instead, they request abstract behaviorally relevant information from EnvInf, such as open space to traverse or a similarity-measure between their stored signature and a currently perceived environment. PAs do not need to understand data provided by the robot and its sensors; they are completely relieved of processing elementary data by receiving behaviorally relevant abstracted information directly from EnvInf.

In summary, the developed environmental representation (EnvInf) collects and maintains available spatial information from multiple on-board sensors and provides behaviorally relevant data. Internally, it memorizes a local environment in a compact log-polar based binned representation. Neither the layer below (a passive robot), nor the layer above (a collection of active place agents) need to inspect or interpret details of stored data.

Creating Individual Places and Building a Network of Places

The data structure described above maintains information about a local, short-range environment. We need to extend the system beyond individual place signatures to allow a robot operating outside its current local environment.

Looking at large environments from a behavioral perspective it is immediately clear that some places are of high relevance, whereas others have hardly any. Highly relevant places provide particular benefits, e.g. a battery charger or a coffee machine; or might require a navigation decision, e.g. turning at an intersection. All such places have different reasons for being relevant, and only the place signature of a particular place reveals why it is relevant, what it provides, and how the robot can behave at that place. Therefore, we design representations of relevant places as active entities, implemented as individual programs called “place agents” (PAs). Each of such a PA autonomously maintains local spatial and behavioral knowledge, and directs the robot to perform adequate actions when it is in the place represented by this PA.

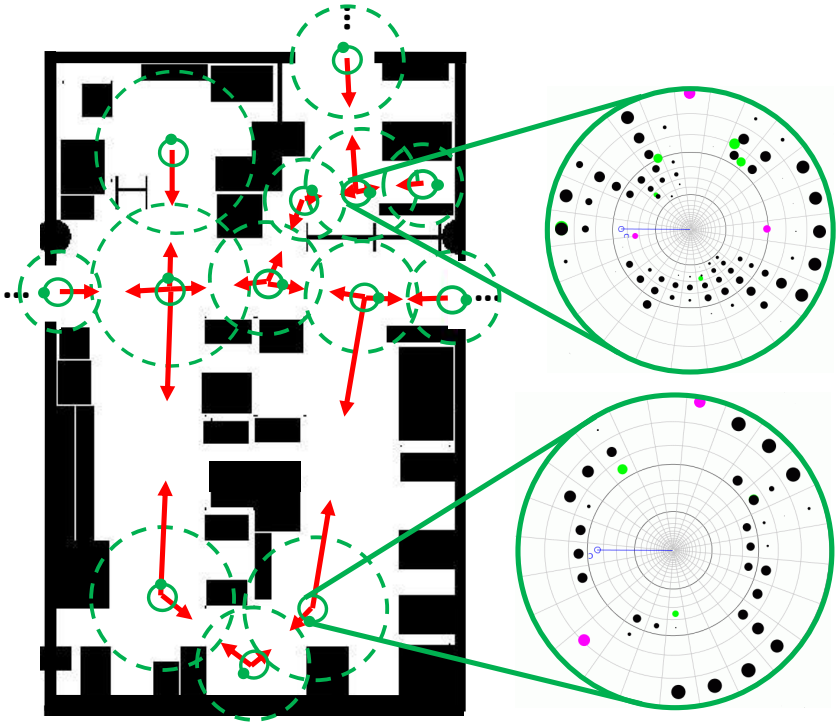

Figure 5: A network of 13 individual place agents (small green circles) representing behaviorally relevant places, superimposed on a plan-view of a room. The large green circles to the right show the internal local spatial knowledge of two example place agents, including pink dots that represent their direct neighbors. Positions of PAs are arranged to reveal the underlying spatial structure, but only for this display; the system is unaware of the PAs’ global spatial arrangement.

Initially, the system starts without any spatial knowledge, with the robot at an unknown position in an unknown environment. It creates an initial PA based on the current place signature reported from the robot. This nucleus PA inspects its local environment for directions in which to continue exploration and creates neighboring incomplete PAs in those directions that we call “blast-PAs” in reference to undifferentiated neuronal blast-cells. As soon as one of these blast-PAs gains control of the robot, it explores its environment while searching for further behaviorally significant occurrences. Detecting such an occurrence, the blast-PA captures a current place signature, remembers the local behavioral relevance, and transforms itself into a full PA. As its parent has

done, the new PA inspects its local environment for unexplored space and creates blast-PAs that continue exploring to find relevant places. Iterating this process creates a collection of individual PAs representing behaviorally relevant places (Figure 5, shown as small green circles). Each PA determines for itself why it is relevant, what it can provide, and what it needs to remember.

In addition to a local place signature, each new PA knows the distance and direction to its parent PA, because it actively guided the robot from its parent's to its current position. Parent-PA and child-PA both know about each other and they know each other's spatial position in their respective coordinate frame. Only this knowledge, distributed among all individually existing PAs, defines a network of PAs (Figure 1, right, and Figure 5). This network is not maintained by a global inspector; rather it exists because of individual nodes maintaining local neighborhood relationships. There is no single actor in the system that contains or has access to knowledge beyond its direct neighbors.

Performing Behaviors with a Distributed Network of Places

Nodes in the network can exchange messages with their direct neighbors. Such messages allow an individual node to coordinate actions that reach beyond its local knowledge. For example, a place agent currently controlling the robot (PA_C) might want to send it to one of its neighbors (PA_N). It rotates the robot towards PA_N within its own local coordinate frame, and notifies PA_N by sending a message. Upon receiving that message, PA_N takes control of the robot, expecting it to arrive from its neighbor PA_C. By constantly comparing the robot's current view with its local place signature, PA_N computes driving commands that attract the robot towards PA_N's center.

Most behaviors, such as fetching coffee, typically reach beyond neighboring places from our current position in the world; but the current PA does not have any information about the world beyond its place and its direct neighbors, neither does it have a reference to "coffee". We have implemented two different approaches to allow globally consistent actions without violating constraints of operating on local information and using local communication only.

The first method allows finding a possibly existing single target anywhere in the network quickly. Such a search is e.g. required whenever a new PA is created during exploration, since the new place might already be represented by another PA elsewhere in the network - in which case the new and old PA merge their spatial knowledge into a single PA. Starting such a search, the current PA_C sends a message to all its neighbors, providing its spatial knowledge (Figure 6, left, red arrows). Each PA receiving such a message from one of its neighbors PA_N adds PA_N's direction and distance to a vector maintained inside the message. This position vector accumulates traveled distance and direction, continuously pointing to an estimate of the position of the origin (PA_C). This estimate is incrementally build-up by adding local distances between neighboring PAs in the local PA's frame of reference. After updating the message's position vector, all PAs propagate the message to all their neighbors, until eventually all PAs in the network receive the message. Each PA evaluates if its own position is sufficiently close to the origin position based on the message's position vector, and compares the received spatial signature with its own local place signature. Upon passing a similarity threshold for spatial and environmental distance, a successful PA sends a reply message along the shortest path to the origin PA (Figure 6, left, blue arrows), identifying itself to represent the same place. Upon receiving such a reply, the origin PA can trigger a fusion process of the two spatial representations for the same place.

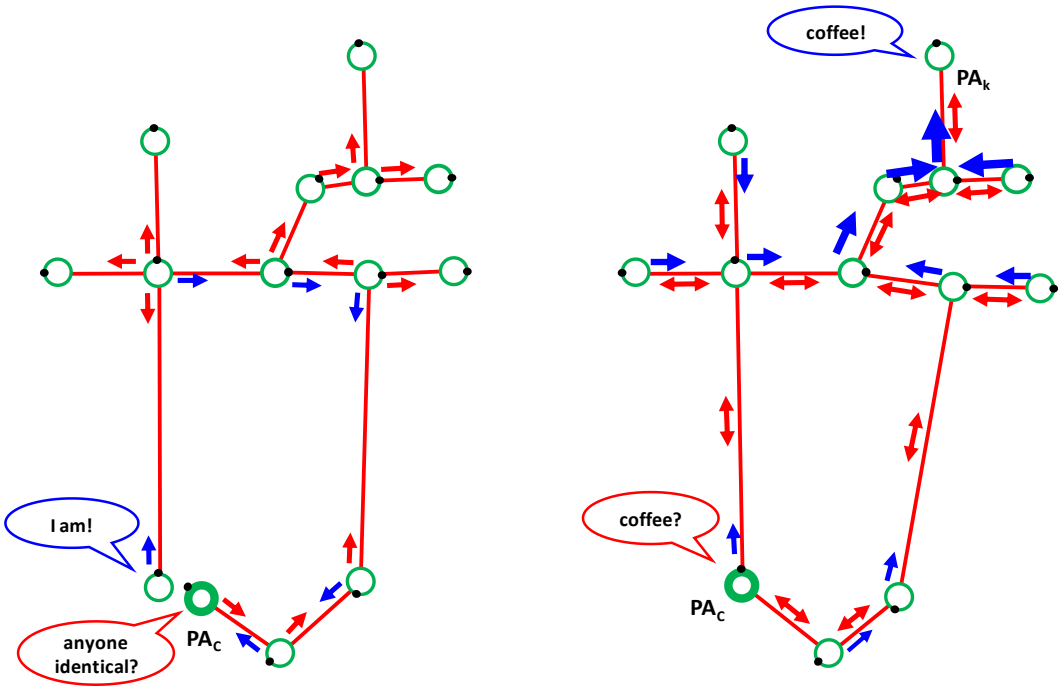

Figure 6: Two different concepts for globally consistent actions: in the left sketch, an outbound message (red arrows) traverses the network searching for an identical place. Upon receiving such a message, each PA computes similarity based on spatial signature and traveled distance. A potentially matching PA returns a message (blue arrows) identifying itself. Alternatively, in the right sketch, an initial "search for coffee" explores the network (red); but after reaching the target PA_K it establishes a gradient towards PA_K (blue), denoted by the size of arrows. Each PA can use this gradient to guide the robot one step closer towards the target. Both methods exclusively use information that is locally available at the time a message arrives; they do not build up global knowledge such as a full path to target or origin.

An alternate method for finding distant places is shown in Figure 6, right, where all place agents establish a gradient towards a distant target. In this scenario, the currently active PA_C sends a "request for coffee" to its neighbors, which immediately reply with their current best knowledge about the request. Initially they have no information, but propagate the request to their neighbors. Ultimately, the kitchen PA_K receives the request and replies "available here" - signaled by distance-to-target of zero meter. Receiving this new information, the kitchen PA's neighbors compute their respective distances to target, based on their knowledge about neighbor PA_K. Having gained new information about the request, the kitchen's neighbors share their updated knowledge with all their further neighbors, which themselves compute accumulated distance to coffee and memorize which of their neighbors provided this information. Repeating the process, eventually all PAs in the network know their respective distance and next-step-neighbor towards coffee, creating a gradient towards the target that each PA uses to direct the robot. Note that the gradient does not operate on absolute direction to coffee, but only maintains neighborhood relations within the network towards coffee. Multiple sources for a request (such as a "Starbucks" besides the kitchen) generate multiple attractors. Using this method, after some time each place agent has insight in how to get closer to a target, even when it is not part of the currently required trajectory. All place agents only use locally available information at the time a message arrived to compute the gradient.

We have implemented diverse complex behaviors, such as inspecting the whole network, searching a previously recorded stimulus or exploring unknown places, all following these simple principles of exchanging messages between local neighboring nodes.

Performance

In contrast to metric approaches that represent a spatial environment (Figure 1, left), such a network of active PAs (Figure 1, right) only represents the vicinity of behaviorally significant places, therefore substantially reducing memory and computing requirements. We implemented the system described above in Java, using a real mobile robot exploring our research institute as an example real-world environment. The system starts out with zero spatial knowledge and autonomously creates a distributed collection of PAs as individual Java processes. On a current computer (2.4GHz AMD quad-core CPU, 4GB main memory) the system performs in real time, allowing the robot to explore an office of 16x7m in about two hours. Simulated tests exploring larger spaces, including the whole institute of about 60x23m, show that the system performs well.

The resulting network of PAs (Figure 2) clearly reflects the behavioral significance of places in the institute: intersections, corners, and dead-ends are represented by a PA. Additionally, artificially labeled places of relevance, e.g. those providing coffee or chocolate, get represented by a PA, allowing the robot to return to these. Space in-between such behaviorally significant places is not represented in the system.

Discussion

The system outlined above implements a new approach to perceiving the world: instead of interpreting spatial information as a large globally consistent map, we perceive the world as a patchwork of independent behaviorally significant spots that one at a time contribute to global behavior.

Starting with zero knowledge, a nucleus node builds up a distributed network of behaviorally relevant nodes while exploring the world. These nodes are active processes, producing offspring for exploration and taking independent decisions based on limited local information. Using such an approach keeps our system computationally tractable in real-time on current hardware.

In contrast to traditional methods for representing spatial information, a unique advantage of our system is that the network of patches does not need to be globally consistent. Because all decisions are taken on local information, local consistency, e.g. in angles and distances, is sufficient. When the mobile agent returns to a previously visited place and closes a loop, traditional systems have to spend much computational effort in correcting acquired information for global consistency. In our system, in contrast, the loop is represented only implicitly, as all nodes only know about their direct neighbors; our system does not require matching angles and distances along the loop. In fact, in case of too high an uncertainty, the nodes can delay the decision to close the loop and temporarily have multiple nodes representing potentially identical places in the environment.

We are currently discussing various extensions of the existing system, ranging from implementations in distributed hardware on-board a small robot, hierarchical structures to represent nodes (such as “a room”, “the Institute”, “the University Campus”), or extending the concepts beyond spatial knowledge (e.g. for a visual system recognizing objects).

Bibliography

1. Arleo, A., *Spatial Learning and Navigation in Neuro-Mimetic Systems, Modeling the Rat Hippocampus*. 2000, Ecole Polytechnique Federale Lausanne: Lausanne.
2. Thrun, S., *Robotic mapping: A survey.*, in *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Editors. 2002, Morgan Kaufmann.
3. Tolman, E.C., *Cognitive Maps in Rats and Men.* The Psychological Review, 1948. **55**(4): p. 189-208.
4. Montemerlo, M. and S. Thrun, *The FastSLAM Algortihm for Simultaneous Localization and Mapping*. Springer Tracts in Advanced Robotics, ed. B. Siciliano, O. Khatib, and F. Groen. Vol. 27. 2007, Berlin/Heidelberg: Springer.
5. Bosse, M., et al., *SLAM in Large-scale Cyclic Environments using the Atlas Framework.* International Journal of Robotics Research, 2003.
6. Shatkay, H. and L.P. Kaelbling. *Learning Topological Maps with Weak Local Odometric Information*. in *International Joint Conference on Artificial Intelligence*. 1997.
7. Tapus, A., *Topological SLAM - Simultaneous Localization and Mapping with Fingerprints of Places*, in *Computer Science and Systems Engineering Department*. 2005, Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland: Lausanne.
8. Kuipers, B.J., *The Spatial Semantic Hierarchy.* Artificial Intelligence, 2000. **119**: p. 191-233.
9. Jefferies, M.E., J. Baker, and W. Weng, *Robot Cognitive Mapping - A Role for a Global Metric Map in a Cognitive Mapping Process*, in *Robot and Cognitive Approaches to Spatial Mapping*, M.E. Jefferies and W.-K. Yeap, Editors. 2008, Springer: Heidelberg/Berlin. p. 265-280.
10. Tomatis, N., I.R. Nourbakhsh, and R. Siegwart. *Simultaneous Localization and Map Building: A Global Topological Model with Local Metric Maps*. in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2001. Maui, USA.
11. O'Keefe, J. and L. Nadel, *The Hippocampus as a Cognitive Map*. 1978, Oxford, UK: Oxford University Press.
12. Redish, A.D., *Beyond the Cognitive Map - From Place Cells to Episodic Memory*. 1999: MIT Press.
13. Hafner, V.V., *Robots as Tools for Modelling Navigation Skills - A Neural Cognitive Map Approach*, in *Robot and Cognitive Approaches to Spatial Mapping*, M.E. Jefferies and W.-K. Yeap, Editors. 2008, Springer: Heidelberg/Berlin. p. 315-324.
14. Milford, M.J., *Robot Navigation from Nature*. Springer Tracts in Advanced Robotics, ed. B. Siciliano, O. Khatib, and F. Groen. Vol. 41. 2008, Berlin/Heidelberg: Springer.
15. Sargolini, F., et al., *Conjunctive representation of position, direction, and velocity in entorhinal cortex.* Science, 2006. **312**(5774): p. 758-62.
16. Markus, E.J., et al., *Interactions between location and task affect the spatial and directional firing of hippocampal neurons.* Journal of Neuroscience, 1995 **15**(11): p. 7079-7094.
17. Blancheteau, M. and A.L. Lorec, *Raccourci et détour chez le rat: durée, vitesse et longueur des parcours [Short-cut and detour in rats: duration, speed and length of course].* L'année psychologique, 1972. **72**(1): p. 7-16.