Unified Human-Robotic Societies

Peter Simon Sapaty

Institute of Mathematical Machines and System National Academy of Sciences, Glushkova Ave42, 03187 Kiev Ukraine

Keywords
World dynamics; Human-robotic Systems; Spatial Grasp Technology; Networked Language Interpretation; Self-evolving Patterns

Abstract
Large numbers of robotic facilities have been accumulated worldwide, but existing robots still remain specialized devices rather than intelligent collaborators for humans. To effectively integrate massive robotics into human societies, radically new and much more universal approaches are needed. A semantic level model supported by special pattern-based knowledge processing language is described. It expresses operations and decisions in distributed spaces in a very compact and mobile form, with traditional system organization and management essentially shifted to automatic language interpretation in cooperative networked environments. Communicating SGL interpreters, associated with humans and robots, can form holistic goal-driven teams under unified command and control, where humans and robots, if needed, can seamlessly substitute each other at runtime, during scenario execution.

1. Introduction
The world is changing dramatically for the last decades, with numerous conflicts and crises emerging frequently and everywhere, which include terrorism, ethnic, religious and military conflicts, endless floods of refugees, economy collapses, and so on. To withstand such unfortunate situations, new system ideologies, approaches, and management technologies are desperately needed. Of particular interest and effectiveness may be those allowing for seamless embedment of massive robotics into human societies, with robots taking care of dangerous and critical situations while acting cooperatively with humans and among themselves under global goals and unified control. But in many areas and cases the existing robots still remain as specialized devices rather than full-scale collaborators for people.

The paper is describing a new and quite unusual approach for human-robot integration which is not pursuing and developing further the traditional and overwhelmingly used interoperability [1, 2] ideology and practice, but rather creating a much higher, “over-operability” [3, 4] layer in the form of supreme (i.e. standing above humans and robots) spatial intelligence. This layer expresses top semantics of what should be done in distributed spaces and main decisions to be taken in complex situations.

Under this approach, it becomes extremely easy to assemble any teams with any ratio between humans and robots, which can substitute each other at runtime without interrupting system missions while always preserving global goal orientation and mission capabilities. Expressing complex spatial operations at this level allows us to automate most of organization and management routines for large human-robotic teams, including sophisticated command and control.

*Corresponding author: Peter Simon Sapaty, Institute of Mathematical Machines and System National Academy of Sciences, Glushkova Ave42, 03187 Kiev Ukraine. E-mail: peter.sapaty@gmail.com

Received Jul 24, 2017; Accepted Aug 09, 2017; Published August 25, 2017

Citation: Peter Simon Sapaty (2017) Unified Human-Robotic Societies. SF J Artificial Intel 1:1.

Copyright: © 2017 Peter Simon Sapaty. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.
The paper briefs the related networking technology that can express operations and top decisions in physical, virtual and executive environments regardless of who (humans) or what (robots) should perform them, and in which quantities. This allows us to make an effective implementation in dynamic environments where manned and/or unmanned resources may not be known a priori but rather defined at runtime, depending on circumstances. The rest of the paper also contains practical examples from different fields confirming efficiency and effectiveness of the approach offered.

2. Spatial Grasp Technology (SGT)

Key ideas of the developed ideology and technology suitable for high-level organization and management of advanced human-robotic teams and its possible networked implementation are revealed in brief.

2.1. Self-evolving Spatial Patterns

Within SGT, a high-level scenario for any task to be performed in a distributed world is represented as an active self-evolving pattern rather than traditional program, sequential or parallel, inheriting holistic and gestalt [5, 6] ideas rather than of communicating agents [7]. This also reflects integral style of human thinking and brain activity [8, 9] when directly perceiving complex images as a whole, while treating parts and their sense within this whole rather than vice versa.

The self-evolving pattern written in high-level Spatial Grasp Language (SGL), expressing top semantics and key decisions of the problem to be solved and starting from any point, spatially propagates, grows, replicates, modifies, covers, interlinks and matches the distributed world, as shown in Figure 1.

The self-spreading & matching patterns can create knowledge infrastructures arbitrarily distributed between system components (humans, robots, sensors), as in Figure 2 (where SGL interpreters are shown as universal control modules U). Covered subsequently or simultaneously by same or other patterns with operations and control, these knowledge infrastructures can effectively support distributed databases, command and control, situation awareness, and autonomous decisions, also simulate any other models, both sequential and parallel like, for example, Petri nets or neural networks.

2.2. Spatial Grasp Language, SGL

SGL [10-12], the core of the approach, allows us to directly move through, observe, and make any actions and decisions in fully distributed environments (whether physical, virtual, executive, or combined). It has universal recursive structure, shown in Figure 3, capable of representing any parallel and distributed algorithms operating over spatially scattered data or other, lower level, distributed systems of arbitrary natures.
SGL main features are in brief as follows.

An SGL scenario develops as parallel transition between sets of progress points (or props), with self-modified and self-replicating scenario code freely moving in distributed spaces. Starting from a prop, an action may result in new props (which may be multiple) or remain in the same prop. Each prop has a resulting value, which may be arbitrarily complex, and resulting state (one of: thru, done, fail, and abort). Different actions may evolve independently or interdependently from the same prop, splitting and parallelizing in space. Actions may also spatially succeed each other, with new ones applied sequentially or in parallel from all props reached by the previous actions.

Elementary operations can directly use states and values of props reached by other actions whatever complex and remote they might be. Any prop can associate with a position in physical, virtual, executive or combined world, sharing local information at them. Staying with the world points, it is possible to directly access and impact local world parameters in them, whether virtual or physical.

Overall organization and control of the breadth and depth space navigation and coverage is provided by SGL rules, which may be nested and can, for example, be as:

- Elementary arithmetic, string, or logic operation.
- Hop in a physical, virtual, execution, or combined space. Hierarchical fusion and return of (remote) data.
- Distributed control, both sequential and parallel.
- A variety of special contexts for navigation in space influencing embraced operations and decisions.
- Type or sense of a value or its chosen usage, guiding automatic interpretation.
- Creation or removal of nodes and links in distributed knowledge networks.
- A rule can be a compound one, integrating a number of other rules; it can also be defined in a result of local or global operations of arbitrary complexity.

Working in fully distributed physical, virtual, or executive environments, SGL has different types of variables, called spatial, effectively serving multiple cooperative processes:

- **Heritable variables** – these are starting in a prop and serving all subsequent props, which can share them in both read & write operations.
- **Frontal variables** – are an individual and exclusive prop’s property (not shared with other props), being transferred between the consecutive props and replicated if from a single prop a number of other props emerge.
- **Environmental variables** – are accessing different elements of physical and virtual words when navigating them, also a variety of parameters of the internal world of SGL interpreter.
- **Nodal variables** – allow us to attach an individual temporary property to different world nodes, accessed and shared by all activities currently associated with these nodes.

These types of variables, especially when used together, allow us to create spatial algorithms working in between components of distributed systems rather than in them, allowing for flexible, robust, and self-recovering solutions. Such algorithms can freely replicate, spread and migrate in distributed environments (partially or as an organized whole), always preserving global integrity and overall control.

To simplify SGL programs, traditional to existing programming languages abbreviations of operations and delimiters can be used too, substituting certain rules as in the examples throughout this text, but always remaining within the general syntactic structure shown in Figure 3.

### 2.3. Networked SGL interpreter

The interpreter (its architecture stemming from [13], more in [14- 16]) consists of a number of specialized modules handling and sharing specific data structures, as in Figure 4.

The SGL interpreters can communicate with each other, and the distributed network of the interpreters can be mobile and open, changing the number of nodes and communication structure in between at runtime. A backbone of the distributed interpreter is its spatial track system providing global awareness and automatic C2 over multiple distributed processes, also creating and managing distributed information and control resources. The distributed SGL interpreter may have any number of nodes, up to millions even billions, spread worldwide.
The dynamically networked SGL interpreters extended by and integrated with other facilities and gadgets, like mobile robots or human-wearable devices, can form universal spatial machines operating with both information and physical matter. These networked machines, working without any central resources under intelligent scenarios injected at any time and from any nodes, can perform complex computational, knowledge processing and control operations.

3. Elementary Examples

- Assignment of the sum of three values 27, 33 and 55.6 to a variable named Result, as in Figure 5:
  \[
  \text{assign(Result, add(27, 33, 55.6))}
  \]

- Move physically from the current location independently and simultaneously to locations (x1, y3) and (x5, y8), see Figure 6:
  \[
  \text{branch(move(location(x1, y3), move(location(x5, y8)))}
  \]
  Will cause movement from the current physical position to the two new physical positions by given coordinates independently and possibly in parallel (if the latter supported by implementation). A shortened version may be as follows:
  \[
  \text{move(x1, y3), move(x5, y8)}
  \]

- Creation of a virtual node Peter, see Figure 7:
  \[
  \text{create(node(Peter))}
  \]
  Starting from the current world location, a new, isolated, virtual node with the given name will be created with the resultant control moving into it. Shortened version:
  \[
  \text{create(Peter)}
  \]
  Extending the virtual network (already having node Peter) with a new link-node pair stating that “Peter is father of Alex”, see Figure 8:
  \[
  \text{advance(hop(node(Peter)), create(link(+fatherof), node(Alex)))}
  \]
The scenario first directly hops into the already existing node Peter and from it creates new link-node pair with both link and node properly named, where the succession in virtual space is provided by the rule advance. Simplified version:

```
hop(Peter); create(+fatherof, Alex)
```

- Giving a command to soldier John to use robot Shooter to fire by coordinates \((x, y)\) with confirmation of the robot’s success or failure, see Figure 9.

```
hop(John);
report_if((hop(Shooter);fire(x,y)), success, failure)
```

### 4. Integral Human-Robotic Teams

In the last example above we showed selective tasking of a human and a robot, whereas will consider here simple scenarios for mixed teams with humans and robots having equal status, as symbolically shown in Figure 10.

#### Randomized group movement, starting in any node, with Range distance allowed between units when moving; units reporting individually if “aliens” seen.

```
hop(all);
node(Limits = (dx(0,8), dy(-2,5)), Range = 200, Shift);
repeat(
  if(seen(unknown), report(‘alien’));
  Shift = random(Limits);
  if(empty(Shift, Range), WHERE += Shift);
  sleep(delay))
```

- Starting from any node, finding topologically central unit of the moving group and hopping into it.

```
frontal(Aver) = average(hop(all); WHERE);
min_destination(hop(all); distance(Aver, WHERE))
```

- Creating hierarchical infrastructure from the center found using oriented links infra and depth as certain maximum allowed linking distance:

```
repeat_linkup(+infra, firstcome, depth)
```

- Using the created infrastructure, collect at its top and analyze all objects (symbolically: targets) discovered throughout the whole territory covered by the group, issuing OK or alarm if danger.

```
frontal(Seen) = repeat(
  free_detect(targets), hop(+infra));
if(analyze(Seen), out(OK), out(alarm))
```

Integration of the above four cases within a single united scenario allowing the whole group randomly move while keeping threshold distance between units, regularly redefining its changing center and hierarchical infrastructure stemming from it, and collection and analysis of targets is trivial – can be done starting from any human or robotic unit (a related case in [17]). Any other collective scenarios can be generated too, often on the fly.

### 5. Fully Semantic Scenario in SGL

At this highest level, it is possible to describe in SGL only what should be done in a distributed space and
which top operations and decisions to make, like follows:

Evaluate damage after disaster in points with physical coordinates $X_1\_Y_1$, $X_2\_Y_2$, and $X_3\_Y_3$, and report the maximum one.

The SGL expression will be:

$$\text{report\_max\_assess}(X_1\_Y_1, X_2\_Y_2, X_3\_Y_3)$$

This semantic description is fully formal, and can be automatically implemented in physical space by available manned, unmanned or mixed units. The solution by robotic units $R_1$ and $R_2$ and manned $M_1$, scattered somewhere in the region (all having communicating SGL interpreters installed) is shown in Figure 11.

Figure 11: Automatic Solution of Semantically Defined Problem

6. Coastal Waters Cooperative Patrol

This is another scenario example, where manned and unmanned units can work together cooperatively and be substituted by each other at any time, with new ones involved too, as in Figure 12, following coastline in changing directions and regularly reporting if discover (sensors dependent) “aliens” in key points.

Figure 12: Simultaneous Coastal Patrol

At the beginning we will create a discrete coastal map as a semantic network consisting of coordinates of key points linked with each other by oriented links (all named $r$). Vehicles will follow this chain along or opposite orientation of the links, changing direction at the end or when see a “colleague” ahead, with the scenario oriented on starting simultaneously in points $x_1\_y_1$, $x_5\_y_5$, $x_9\_y_9$.

$$\text{stay\_create}(x_1\_y_1;(+r,x_2\_y_2);...;(+r,x_9\_y_9));$$

$$\text{hop}(x_1\_y_1,x_5\_y_5,x_9\_y_9);$$

$$\text{frontal}(R) = \text{random}(+r,-r)$$

WHERE = CONTENT;

$$\text{repeat}(\text{repeat}(\text{check\_report}(...));\text{none}(\text{distance});\text{invert}(R))$$

This semantic level scenario can, for example, be executed by unmanned UPV1 and UPV2 vehicles and manned MPV1, as in Fig. 12. In case of a manned vehicle engaged, the boldfaced operations can be performed manually, whereas in robotic cases – all automatically.

7. Swarm against Swarm Aerial Scenario

We will consider here the case where a manned, unmanned or mixed aerial swarm is opposing another group of aerial vehicles, which may be manned/unmanned too. This, for example, can relate to fighting criminal and spying drones which are currently spreading worldwide [18, 19] and may potentially operate in swarms too.

Main roles of the swarm against swarm scenario, with alien drones as Targets and friendly units as Chasers are shown in Figure 13, with SGL scenario description following and explanation of its main steps further down.

Figure 13: Fighting Group Targets with Manned/Unmanned Swarms
This will be working in the following (including parallel) steps, where each time after distribution of all collected targets the fully mobile scenario is starting from another, randomly chosen chaser.

- Initial launch of the swarmed chasers (in Figure 13 with SGL interpreters U embedded, which can communicate with each other) into the expected operational area.
- Discovering targets, finding their topological center, and forming priority list by their physical positions in relation to this center, where highest priority is assigned to topologically central targets as potential control units of the suspected intruders.
- Other targets are sorted by their growing distance from the topological center of the group.
- The most peripheral targets (those in maximum distance from the group’s center), may be assigned higher priority too as potentially having more chances to escape, and being prevented from this.
- Stepwise assigning of available chasers to highest priority targets (for each target the physically nearest chaser is chosen) classifying the chosen chasers as engaged with subsequent individual chasing and neutralizing the targets.
- Restoring status vacant after performing the task if chasers survive themselves.
- The vacant chasers are again engaged in the targets selection & impact.

It is worth noting that all the chaser swarm management has been done exclusively within the swarm itself, by human or artificial intelligence and without external intervention, which can dramatically simplify an outside group tasking and potentially involve any number of collectively behaving manned or unmanned units.

8. Conclusions

The current paper pioneered on formalization of semantic level operations and top intelligence as regards large distributed systems, which can be implemented by any available resources regardless of being human or robotic, thus paving a real way to integration of multiple robotics into human societies.

Some remotely related works in this direction are conducted in military on formalization of Command and Control (C2) to simplify multilingual international cooperation and also improve chances of formal engagement of robotic facilities in advanced operations. But the developed specialized Battle Management Languages (BML) [20] for unambiguous expression of C2 are not programming languages themselves, therefore needing integration with other linguistic facilities and organizational levels. On the contrary, SGL, being fully formal and universal system language, allows for effective and compact semantic expression of any battlefield scenarios and orders, also directly supporting robotized up to fully robotic systems [17]. More on SGL, its history, applications, and international cooperation can be found elsewhere, [21-24] including.

The latest and most advanced version of the
technology can be put on any platform in a short time and by a small group of system programmers, within existing university environments too. The author would be glad to communicate with organizations and individuals who may get interested in this area of research and cooperation.

References

1. Interoperability, Wikipedia.


