An Evolutionary Algorithm for Manipulator Path Planning

Ralf Corsépius

Research Institute for Applied Knowledge Processing (FAW)
Helmholtzstr. 16
89081 Ulm, Germany

corsepiu@faw.uni-ulm.de

Abstract

In this paper, a versatile and scalable manipulator path planning algorithm based on an evolutionary algorithm will be described. The evolutionary algorithm realizes path planning by probabilistically searching feasible solutions in configuration space, with evolutionary rating taking place both in configuration space and work spaces simultaneously. This algorithm is intentionally kept simple and avoids inverse kinematics, which allows it to be applied for path planning of high-dimensional and redundant manipulators.

1 Motivation

Applications of manipulator on mobile robots (*mobile manipulators*) differ basically from conventional applications of stationary manipulators as they can be met, for instance in industrial applications. While path planing of manipulators there often is restricted to typically few, pre-programmed, highly accurate paths in a single, well known, often static environment, mobile manipulators impose demands to act in different, changing, and sensorially perceived work spaces under near real-time conditions.

Nowadays, such kind of systems are hardly implementable, because the necessary technical pre-requisites can only be partially met, resp. because necessary algorithms are not (yet) available. For practical applications of mobile manipulators, therefore, manipulation processes often are restricted to simple operations, for instance to parameterized paths, in more or less structured environments, as well as to simple manipulators with very much restricted manipulation possibilities and low numbers of degrees of freedom.

One possibility to realize more complex manipulation operations, is to apply manipulators with higher degrees of freedom. This implies the necessity to be able to perform actorical and planning tasks on-line and near real-time, i.e. from within a few seconds up to a couple of minutes. This work tries to contribute its share towards this direction in front of the background of mobile manipulators.

2 Introduction/Considerations

Path planning has been a central topic to robotics for many years. For an overview of common algorithms please refer to Latombe [1] and other works referring to it, such as [2],[3], [4] and [5], only to mention a few.

2.1 Configuration Space- / Work Space Planner

First of all, one can distinguish between configuration space and work space based path planners. While the former perform their underlying computations in *joint space* (or *configuration space*), the latter are based on computations in *work space* (Transformation of the configuration space into Cartesian space) of the robot. The transformation from configuration space into work space is called *direct kinematics*, their inverse is the so-called *inverse kinematic*s.

While the direct kinematics of a manipulator typically is solvable by simple and analytical means, the solution of the inverse kinematics is much more complex, because it is not always possible to find a closed-form solution, multiple solutions may exist, infinite solutions or no admissible solutions may exist [6]. For commonly used manipulator kinematics with less than six degrees of freedom corresponding solutions can be found in literature [7, 6], kinematic systems with more than six degrees of freedom (redundant kinematics) no general closed-form solution of the inverse kinematics is known.

As configuration space planners in principle do not need the inverse kinematics, they are basically better suited for large numbers of degrees of freedom than work space planners. On the other hand, specifying goals in configuration space is much more difficult than in work space, and is the reason why configuration space based planners in practical applications often additionally require the inverse kinematics (e.g. our planner in [4] and [8]).

2.2 Global / Local Planner

Basically, two types of path planning algorithms can be distinguished: Global planners and local planners.

Global planners are typically complete in the sense of being able to find a solution if one exists. Their disadvantage is that the required effort in general grows exponentially with the number of degrees of freedom taken into account for planning. Therefore, such kind of algorithms often can only be applied off-line for systems with a higher number of degrees of freedom.

Local planners restrict their scope of planning to a local neighborhood and therefore in many cases are faster than global planners. However, they return locally valid solutions, only, which not necessarily are identical to global solutions.

It's also characteristic for local planners to get stuck in local solutions, instead of finding the desired (global) solution. I.e. in terms of optimization theory, to return a local minimum instead of the actually desired global minimum. One possibility to approach this problem is to insert solutions found by local planners as support points into graphs and let this graph be evaluated by searching algorithms (z.B. Dijkstra, A^*). Examples for planners applying of such algorithms can be found in Kavraki ([2], road maps), Ahuacztin ([9], Expore-Task) or Braun/Corsépius ([4]).

2.3 Optimization Problem

The path planning problem can also be regarded as the solution of a high-dimensional pareto-optimization problem. Therefore, basically, all known techniques to solve this class of problems can be considered potential candidates to approach the path planning problem.

However, many classic optimization techniques, such as dynamic programming, fail to be applicable for practical applications, in particular to on-line path planing of robotic systems, due to their high complexity (often $> O(m^n)$, with m number of discretization steps per degree of freedom and n number of degrees of freedom) and their resulting high demands on memory and/or computational power.

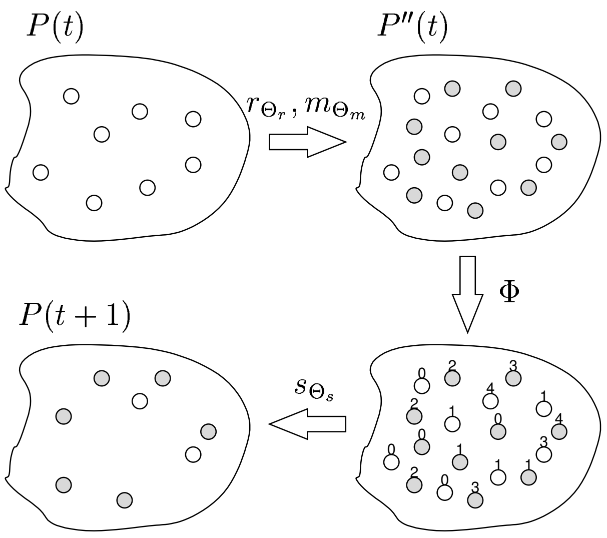

Figure 1: Life cycle of the population of an evolutionary algorithm

If interpreting path planning as a high-dimensional search problem, evolutionary algorithms [10] are a natural candidate, because, due to their probabilistic character, they promise to have the ability to quickly find solutions in a large number of high-dimensional problems.

3 Evolutionary Algorithms

The term *evolutionary algorithms* has been introduced in newer literature [11, Baeck96]as a generic term to denote a family of popular algorithms, which have been developed independently since the beginning of the 1960s.

Essentially these are:

- Genetic Algorithms (Fraser, Holland [12], Goldberg)
- Evolutionary Programming (L.J. Fogel, D.B. Fogel)
- Evolutionary Strategies (Bienert, Rechenberg, Schwefel)

All of them share the same basic structure depicted in algorithm 1 and figure 1, however differ in details and especially in their history of development [10].

Common to all is a population $P(t)$ of individuals $\vec{a}_i$ which is subject to a cycle that is modeled after evolution of biological populations:

Algorithm 1 General evolutionary algorithm (BÃd'ck[10])

```
t := 0
initialize $P(0) := \{\vec{a}_1(0), ..., \vec{a}_\mu(0)\} \in I^\mu$;
evaluate $P(0) : \{\Phi(\vec{a}_1(0)), ..., \Phi(\vec{a}_\mu(0))\}$;
while $\iota(P(t)) \neq$true do
    recombine:$P'(t) := r_{\Theta_r}(P(t))$;
    mutate:$P''(t) := m_{\Theta_m}(P'(t))$;
    evaluate $P''(t) : \{\Phi(\vec{a}''_1(t)), ..., \Phi(\vec{a}''_\lambda(t))\}$;
    select: $P(t+1) := s_{\Theta_s}(P''(t) \cup Q)$;
    $t := t + 1$;
end while
```

Throughout this cycle, from an existing population $P(t)$ a new population $P''(t)$ is generated by means of recombination $r_{\Theta_r}$and mutation m_{Θ_m}. Individuals of this new population then are undertaken an evaluation Φ. Unified with a set Q of spontaneously and randomly added individuals, a selection algorithm s_{Θ_s}selects a new initial population $P(t+1)$ of individuals for a new evolution cycle. The algorithm terminates if one or several individuals met a certain termination criterion ι.

4 Approach

The basic idea of this approach is to treat the posture of a manipulator in its configuration space as an individual of an evolutionary population, which is subject to an evolutionary process and gets evaluated with respect to fitness-criteria from both work- and configuration space (section 3).

4.1 Problem

Given:

1. The momentary posture of a manipulator in configuration space:
$$\overrightarrow{\varphi} = (\varphi_1 ... \varphi_n)$$
2. The forward kinematics of a manipulator, given as kinematic chain, denoted as chain of homogeneous transformations (Denavit-Hartenberg transformations [13])
$${}^0T_n(\overrightarrow{\varphi}) = {}^0T_1(\varphi_1) * {}^1T_2(\varphi_2) * ... * {}^{n-1}T_n(\varphi_n)$$
3. Posture of a goal object in workspace coordinates, denoted as homogeneous transformation:
${}^0T_{goal} = {}^0T_{goal}(\overrightarrow{x}) =$
${}^0T_{goal}(x_{goal}, y_{goal}, z_{goal}, \alpha_{goal}, \beta_{goal}, \gamma_{goal})$

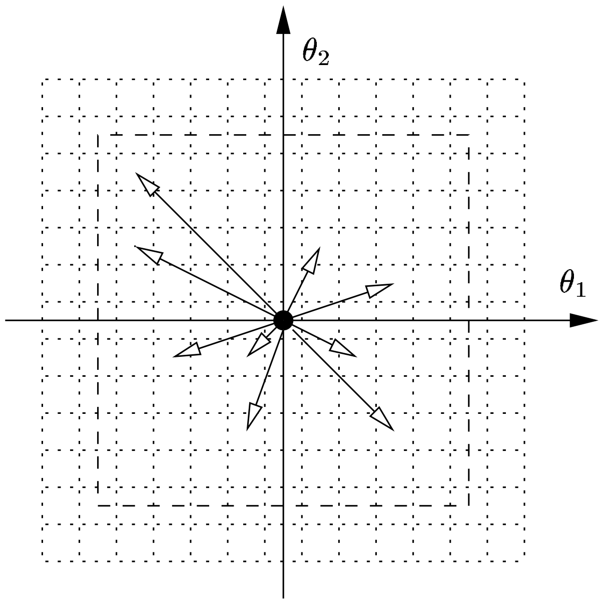

Figure 2: Local search scope of an individual in configuration space.

Wanted is the posture $\overrightarrow{\varphi}$ in configuration space of the manipulator, which minimizes a fitness function $f(\overrightarrow{\varphi}, {}^0T_n(\overrightarrow{\varphi}), {}^0T_{goal})$.

4.2 Evolution Strategy

The evolution strategy having been developed is a specialization of the algorithm depicted in figure 3.

As initial population $P(0)$ the momentary posture $\overrightarrow{\varphi}$ of the manipulator in configuration space will be used (i.e. population size $\mu = 1$). Starting with this single individual, in an initialization phase, a new population $P''(1)$ is generated by random local variations of the components of the posture vector $\overrightarrow{\varphi}$ (typical population size $\mu = 50$).

Each single individual from population $P''(t)$ then will be evaluated by fitness functions, which perform a combined rating of an individual's posture, applying different configuration and work space criteria, simultaneously.

Subsequently, in a selection step, a subset of individuals (typical size 5), carrying best fitness values (rating) of the population, will be chosen as new initial population for the next evolutionary cycle.

The algorithm terminates if an individual of the evolutionary population matches a given success criterion (e.g. sufficiently small distance between end-effector ${}^0T_n$ and goal ${}^0T_{goal}$) or if no improvement of the fitness of the population's best individual could be achieved within a given number of cycles (aborting planning).

The desired path results from the accumulated history

$$\vec{\varphi}(t=0), .., \vec{\varphi}(t=t_{end})$$

of the winning individual.

4.3 Fitness Functions

To evaluate a single individual, almost arbitrary configuration- and/or work-space fitness functions and combinations of them can be applied. By combining different fitness functions and in combination with different termination criteria, different elementary behavioral patterns/manoeuvres of the manipulator, such as "approaching", "detaching" and can be implemented.

4.3.1 Fitness Functions in Configuration Space

Fitness functions in configuration space solely rate the current values of the manipulator's posture $\vec{\varphi}$ in configuration space. They can be applied to include joint angle limitations into planning.

Proven to be useful for this purpose have binary ratings ("Angle is within bounds") and rating the minimal distance of the current joint angle value to its bounds.

4.3.2 Fitness Functions in Work Space

1. *Comparison of the end-effector posture* ${}^0T_n$ *against the posture of the goal object* ${}^0T_{goal}$: Here, the basic idea is to have specific ratings attached to goal-objects, which can be applied to realize different manoeuvres for specific types of goal objects. For example:
   - A general, universally applicable approach manoeuvre can be realized by rating the Euclidian distance between goal objects and the end-effector.
   - For cylindrical goal objects, axial approach manoeuvres can be realized by rating the difference between a vector perpendicular to the end-effector's main grasping direction and the main axis of the cylinder.
2. *Rating the end-effector posture* ${}^0T_n$: These can be applied to let the end-effector prefer certain regions of the work space during planning. For example, one can implement a rating of the rotation sub-matrix of ${}^0T_n$ to restrict motion of the end-effector (e.g. parallel to the ground plane)
3. *Distances between objects in workspace*: In first place, they serve collision avoidance between objects in workspace, comprising collisions between the links of the manipulator and against the environment and between individual links of the manipulator. In second place, ratings based on distances in work space can be applied to let the manipulator prefer free regions of the work space instead of "scratching along surfaces" ("Free-space/obstacle-avoidance" behavior).

4.3.3 Combined Fitness Functions

Basically, it is possible to set up arbitrary computation recipes for combined fitness functions. As they depend upon the desired manoeuvre and because each of the fitness functions involved comes with a variety of possible choices of parameters, setting up combined fitness functions is not an easy task. Here, we have restricted ourselves to implementing a few elementary manoeuvres which are useful in our applications.

For example, a simple form of *approaching an object* can be implemented this way:

- Minimizing the distance of the end-effector to the goal object by rating the Cartesian distance from end-effector to goal object surface,
- Obstacle avoidance by applying a binary rating of distances between all manipulator links to the environment.
- For joints with bounds, additionally add a rating for nearness to the boundaries, favoring maximal distance to the bounds.

5 Results

The approach described herein is work in progress. It currently is subject to simulation studies covering different manipulator kinematics (cf. figure 3) in different workspaces with and without obstacles. Typical testing scenarios comprise "pick-and-place" service-robotics tasks in artificial, semi-structured, prototypical service robots environments, for example, "picking up a bottle from a table with a MANUS manipulator"

For these simulations, a software system, based on hierarchical, object-oriented, three-dimensional, geometric-topological, kinematic world models [14, 15] has been

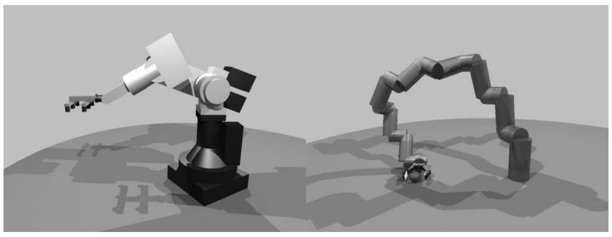

Figure 3: Examples for simulation models: Left industrial manipulator, right ”snake-like” manipulator (16 DOF)

developed that allows simulation of manipulators and their environment, comprising 3D-animation (Open-Inventor) and collision-detection/distance-computation ([16, 8]) under near real-time conditions on standard (Pentium III class) PC-hardware.

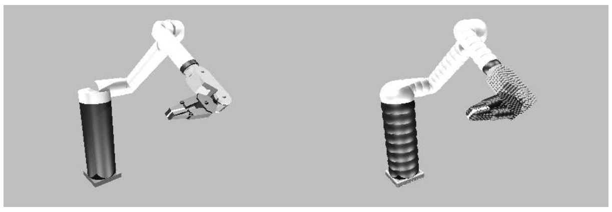

Figure 4: Model of a MANUS manipulator (left: Geometric Model, right: internal model applied for collision-/distance computation).

The simulation models internally consist of a topological tree of attributed real world objects (e.g. bottle, ball, specialized robot links etc.). For collision detection/distance, each of these objects, is transformed into a set of geometrical primitives (cube, cylinder etc.) and, in a subsequent step, into sets of spheres (cf. figure 4), representing an approximation of the surface of the real world object (Typical number of spheres:10^5-10^6; Fig. 4 consists of ca. 30000spheres).

The typical size of the evolutionary population having shown to be reasonable is 100 individuals. Simulations indicate that using larger population sizes don’t necessarily improve planning speed (Sparse search, ”needle in the haystack”). Using smaller sizes overproportionally increase planning times and reduce convergence.

Due to the probabilistic nature of the approach, deterministically reproduceable planning times can hardly be specified. For simple tasks/scenarios, such as approaching a ball with the 6-DOF MANUS manipulator in free work space, planning times can range from a couple seconds up to a few minutes.

The computational effort, and therefore the required planning time is higher than that of recent fast path planning algorithms, such as[5] and [4]. For practical applications however this disadvantage can be at least partially compensated by the scalability of the algorithm (Number of individuals in a population μ, size of the local search scope during generation of an individual).

6 Conclusion

A simple path planning algorithm for manipulators has been developed, which on one hand is based on combination and extension of known algorithms [3, 1, Baeck96], and their directed adaptation to a specific application, on the other hand hereby avoids typical problems of conventional path planning algorithms.

Worth emphasizing seems to be the fact that inverse kinematics can be avoided, because the actual planning takes place in configuration space, while rating of reachable intermediate goal can take place in work space as well as in configuration space (solving the inverse kinematics implicitly and iteratively).

This also allows to take numerous constraints into consideration for planning (e.g. grasped object may not be tilted), which are hardly integrable into other path planning algorithms, especially to pure configuration or work space based planners.

Furthermore, this allows us to apply this approach to highly-redundant manipulator kinematics, and enables us to realize more complex manipulation manoeuvres.

Like other local path planners, this algorithm shows has the characteristic of potentially ”getting stuck” in local minima, i.e. convergence against a global solution of the path planning problem can not be guaranteed. Here, combinations with other global path planning algorithms, in particular graph based algorithms, seem to offer potential enhancement opportunities [2, 17, 4].

The presented algorithm is a first realization of this approach and leaves open a number of possibilities for further development and improvement. The evolutionary algorithm has been kept simple on purpose, and we therefore refrained from applying further standard methods of evolutionary algorithms, such as Cross-Over, Gaussian distributed selection of individuals from their parent population when generating a generation, Gaussian distributed mutation rates around a parent individual etc.

Furthermore, developing a generally applicable, combined fitness function to implement generally applicable, reasonable behavioral patterns has proven to be difficult.

Acknowledgements

Parts of the work described herein are based on work from project AMOS (granted by BMBF), which had been continued and developed further in the Special Research Area (SFB) 527 "Integration of symbolic and subsymbolic information processing in adaptive sensomotoric systems" (granted by the Deutsche Forschungsgemeinschaft, DFG).

References

[1] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Press, 1991.

[2] L. E. Kavraki, "Random networks in configuration space for fast path planning," Ph.D. dissertation, Stanford University, 1994, ftp://flamingo.stanford.edu/pub/kavraki/thesis1.ps.

[3] S. Quinlan, "Real-time modification of collision-free paths," Ph.D. dissertation, Stanford University, 1994.

[4] B. Braun and R. Corsépius, "Amos: Schnelle manipulator-bewegungsplanung durch integration potentialfeldbasierter lokaler und probabilistischer algorithmen," in *Autonome Mobile Systeme, 12. Fachgespräch*. München: Springer, 1996.

[5] B. Baginski, "Motion planning for manipulators with many degrees of freedom – the bb-method," Ph.D. dissertation, Fakultät für Informatik, TU München, 1998.

[6] L. Sciavicco and B. Siciliano, *Modeling And Control Of Robot Manipulators*. McGraw-Hill International Editions, 1996.

[7] C. Canudas de Wit, B. Siciliano, and G. Bastin, *Theory of Robot Control*. Springer, 1996.

[8] R. Corsépius and B. Braun, "Amos: Fast manipulator path planning by integrating potential based local methods and probabilistic global planning," in *MCPA '97, 2nd Int. Workshop on Mechatronical Computer Systems For Perception And Action*, G. Buttazzo and E. Ricciardi, Eds. Pisa/It: Scuola Superiore di Studi Universitari S.Anna, 1997.

[9] J.-M. Ahuactzin Larios, "Le fil d'ariadne: Une méthode de planification générale. application à la planification automatique de trajectoires," Ph.D. dissertation, LIFIA, Genoble, 1994.

[10] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[11] J. Heitkötter and D. Beasley, "The hitch-hiker's guide to evolutionary computation: A list of frequently asked questions (faq)," ftp://rtfm.mit.edu/pub/usenet/news.answers/ai-faq/genetic/, p. ca. 110, 2000.

[12] J. H. Holland, *Adaptation in Natural and Artificial Systems*, 2nd ed. MIT Press, 1975.

[13] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, June 1955.

[14] R. Corsépius, "Amos: World modelling for autonomous mobile manipulators," in *7th ANS Topical Meeting on Robotics and Remote Systems*. Augusta/GA: ANS, 1997.

[15] ——, "Computation of spatial relations for interaction of symbolic and subsymbolic information processing," in *8th ANS International Topical Meeting on Robotics and Remote Systems*. Pittsburgh/PA: ANS, 1999.

[16] S. Quinlan, "Efficient distance computation between non-convex objects," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1994, pp. 3324–3329.

[17] P. Bessière, J.-M. Ahuactzin, E.-G. Talbi, and E. Mazer, "The ariadne's clew algorithm: Global planning with local methods," in *Proc. IEEE Conference On Intelligent Robots and Systems (IROS)*. Yokohama, Japan: IEEE/RSJ, 1993.