SPARTA: Fast Distributed Robot Trajectory Planning

Charles Mathy, Felix Gonda, Dan Schmidt,
Nate Derbinsky, Alex Alemi, Jose Bento,
Francesco Maria Delle Fave, Jonathan Yedidia

The problem

We take a set of N simple robots, i.e. circles in 2D or spheres in 3D, and seek to find smooth trajectories that get them to their destination from their starting points colliding neither with each other nor static obstacles.

We formulate this as an optimization problem. The paths of the robots

$$\vec{r}^{\,a}(t) \quad a = 1, \ldots, N \quad t \in [0, T]$$

are defined from time t=0 to t=T, and we fix the start and endpoints:

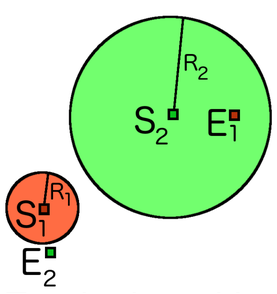
$$\vec{r}^{\,a}(0) = \vec{S}_a\,,\ \vec{r}^{\,a}(T) = \vec{E}_a$$

The robots have radii R_a. We are free to choose an objective function we minimize. Here we first choose smooth desired trajectories $\vec{r}^{\,Da}(t)$ for each robot, e.g. straight lines from start to endpoints. The objective is

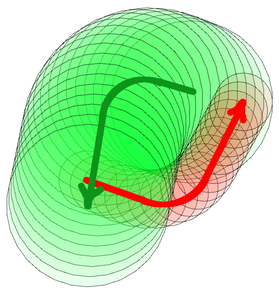
$$f = \sum_{a=1}^{N} \int_0^T dt\ (\dot{\vec{r}}^{\,a}(t) - \dot{\vec{r}}^{Da}(t))^2$$

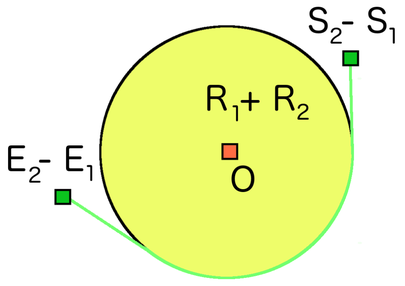
which we seek to minimize with the constraints that all times in [0,T] the robots do not collide with each other or obstacles; the magnitude of their velocities, accelerations and angular velocities are below given values; their velocities are continuous.

To get some intuition for this problem, we can solve the problem of two robots without obstacles or physical constraints exactly. The robots first move in a straight line until they graze each other, then circle around each other until they move in a straight line to their destination.



Two body problem

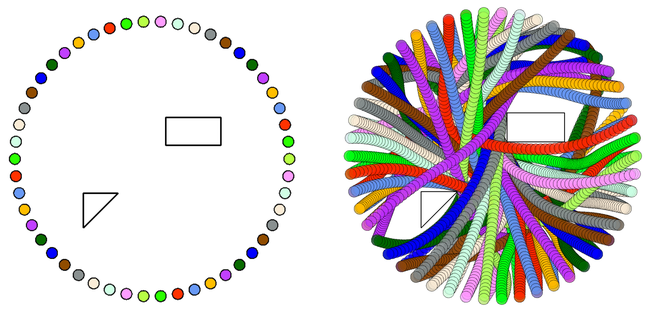

Exact solution



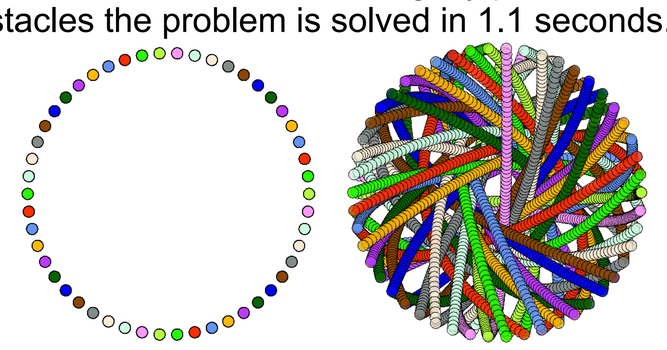
Solution in the frame of robot 1

Tests

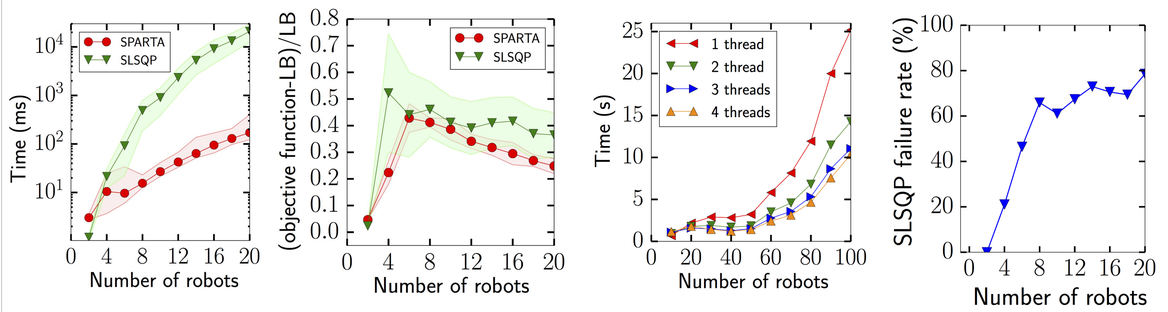
For example, put 50 robots in a circle that have to move to the antipodal point, with some set maximum velocity, acceleration and angular velocity. Using 5 splines to represent each robot trajectory, this problem is solved in 4.3 seconds with 4 threads.



The obtained solutions are smooth and tightly packed. Removing the obstacles the problem is solved in 1.1 seconds.

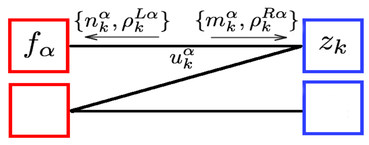


We compare to solving the whole problem with the industrial standard for nonlinear optimization, SQP. SPARTA is faster, more robust (SQP sometimes fails to converge, SPARTA always converges for the robot circle exchange problem), finds better solutions and is parallelizable.



SPARTA formulation

SPARTA is built on a recently developed generalization of ADMM called the Three-Weight Algorithm, which can be formulated as a message passing algorithm. The problem is broken down into factors: the objective function is a sum of factors, and each constraint contributes a factor. We put the factors on the left and variables on the right.

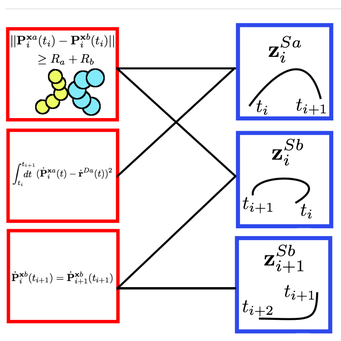


A line is drawn connecting each factor to each variable it depends on. For each factor, we create local copies of the variables it depends on, associated u variables which are Lagrange multipliers,weight variables to the left (right) ρL (ρR), and messages to the left (right) n(m). At each iteration, each factor receives its copy of the variables with associated weights. It solves a subproblem independently of the other factors, which involves minimizing its fα plus a quadratic penalty for differing from the values of the variables it received:

$$x_k^\alpha \leftarrow \underset{x_k^\alpha}{\operatorname{argmin}} \left[f_\alpha(\{x_k^\alpha\}) + \sum_k \frac{\rho_k^{L\alpha}}{2} (x_k^\alpha - n_k^\alpha)^2 \right]$$

After the factors are done, the messages are sent to the right and averaged. For TWA, if a constraint factor is inactive, it sends a zero-weight message (in ADMM all weights are equal). This can significantly reduce convergence time on non-convex problems.

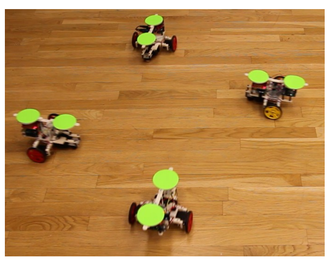
In SPARTA we represent the paths as a fixed number of second order splines. The factors we have are soft objective, robot-robot collision and robot-obstacle collision constraints, physical constraints on velocity, angular velocity, acceleration, and factors enforcing continuous first derivative where the splines meet. Some of the factors can be solved analytically, others are solved with Sequential Quadratic Programming (SQP).

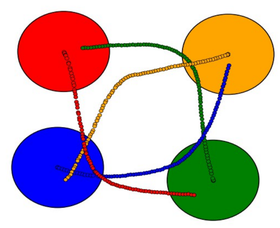


Hardware implementation

To test the real-time capacity of SPARTA we built simple two-wheeled mobile robots, with arduinos controlling servo wheels and xbees for wireless control. A camera tracks position and orientation. Given positions and desired paths, a PID controller sends controls to the wheels.The error margin of the controller (2 inches) is incorporated in the effective robot radius. We can run the system in two modes:

- Setting a desired final trajectory. Here we execute the robot exchange described previously, for 4 robots. At a frequency of 10 Hz we set the measured positions of the robots as the initial positions and the targets as desired final positions. We run SPARTA for 10 ms and feed its output to the controller. The cars do not collide. Tracking takes 5 ms and sending controls another 5 ms, which means we have 70 ms of computation time left per iteration.





- Responding to user inputs in real-time. Here we put 8 robots in a virtual bounding region and let users set a desired velocity and angular velocity. The desired paths are a simple forward prediction based on inputs. SPARTA takes these potentially colliding paths and outputs collision free trajectories. Tracking takes 10 ms for 8 cars,and PID control also takes about 10 ms. We run SPARTA for 20 ms at a frequency of 10 Hz.