Singularity-Crossing Cartesian Motion for a 6-DOF Robotic Arm

Administrative Info

Project: Singularity-Crossing Cartesian Motion for a 6-DOF Robotic Arm
Responsible: Arno Laurie (Software Systems Engineer)
Contact: arno.laurie@epfl-xplore.ch

1 Context and Motivation

EPFL Xplore is a student-led Swiss robotics association, part of the MAKE Initiative at EPFL. Each year the team builds a Mars-analogue rover to compete in the European Rover Challenge (ERC) in Krakow. The rover includes a 6-DOF robotic arm used for tasks such as manipulating switches, inserting probes, operating panels and handling samples.

Many of these operations require the end-effector to follow a Cartesian trajectory. For a 6-DOF manipulator, Cartesian velocity and joint velocity are related by

$$v = J(q)\dot{q},$$

where $v \in \mathbb{R}^6$ denotes the end-effector twist and $J(q)$ is the manipulator Jacobian, expressed using a consistent reference frame and convention. At a kinematic singularity, the Jacobian loses rank. As a singular configuration is approached, one or more singular values of the Jacobian approach zero, causing Jacobian-inverse and pseudoinverse methods to become ill-conditioned and potentially demand very large joint velocities. At the singular configuration itself, some instantaneous Cartesian velocity directions cannot be generated by any finite joint velocity.

However, reaching a singular configuration does not necessarily mean that the desired Cartesian trajectory is physically impossible. For some trajectories a continuous joint-space motion exists before and after the singular configuration. The challenge is to find this continuous motion while keeping joint motion within acceptable limits.

The goal of this project is therefore not to avoid singularities, but to study when they can be deliberately crossed and how trajectory-level inverse kinematics can improve the resulting motion.

2 Technical Background

Standard differential inverse kinematics computes joint velocities from a desired Cartesian velocity using the Jacobian or one of its regularized inverses.

Common approaches include:

- the Moore–Penrose pseudoinverse;
- damped least-squares inverse kinematics;
- adaptive damping near singular configurations;
- sequential pose inverse kinematics;

These methods behave differently when the Jacobian becomes poorly conditioned. Damping can keep joint velocities bounded, but generally introduces Cartesian tracking error. Pose-based IK methods can also switch between different inverse-kinematics branches, producing large and unnecessary changes in joint angles.

Instead of solving the inverse-kinematics problem independently at every instant, this project investigates solving for a sequence of joint configurations associated with an entire Cartesian trajectory. Considering neighbouring

configurations together makes it possible to explicitly encourage continuity and impose joint position, velocity and acceleration limits.

The project does not assume that every singularity can be crossed. An important part of the study is therefore to distinguish successful singularity crossings from Cartesian trajectories that are simply infeasible.

3 Research Question and Objectives

Research question. *Can trajectory-level inverse kinematics allow a non-redundant 6-DOF robotic arm to execute Cartesian trajectories through kinematic singularities more reliably than conventional local inverse-kinematics methods, while respecting actuator limits and maintaining acceptable Cartesian accuracy?*

To answer this, the student will:

1. Study manipulator singularities and existing singularity-robust and singularity-consistent inverse-kinematics methods.
2. Build a simulation framework for a representative 6-DOF robotic arm, including forward kinematics, Jacobians, joint limits and Cartesian tracking-error measurements.
3. Construct deterministic Cartesian trajectories that pass through or close to wrist, shoulder and workspace-boundary singularities.
4. Implement conventional inverse-kinematics baselines such as pseudoinverse IK, damped least squares and sequential waypoint IK.
5. Develop a trajectory-level inverse-kinematics method that searches for a continuous joint trajectory while taking joint position, velocity and acceleration limits into account.
6. Compare the methods in terms of Cartesian tracking error, joint motion, singularity-crossing success and computational cost.
7. Evaluate how the result changes with Cartesian speed, actuator limits and the distance of the trajectory from the singular configuration.

The exact formulation is part of the project and is expected to be refined by the student.

Idea. Rather than solving the inverse-kinematics problem separately at each instant, the whole joint trajectory is solved for at once. Two things are treated as unknown: the joint configurations, and how fast the end-effector advances along the path.

Near a singular configuration the smallest singular value $\sigma_{\min}$ of $J(q)$ tends to zero, and holding a constant Cartesian speed then requires joint velocities growing like $1/\sigma_{\min}$. Slowing down at the crossing keeps them bounded. If the timing is fixed in advance this option is unavailable, and a path that is in fact crossable is reported as impossible. The timing is therefore an unknown of the problem, not an input.

Desired motion. The desired motion is given as a geometric path

$$T_d : [0,1] \longrightarrow SE(3), \qquad s \longmapsto T_d(s) = \big(R_d(s),\, p_d(s)\big),$$

parametrized proportionally to arc length, so that equal steps in s correspond to equal Cartesian distance.

Unknowns. The motion is discretized into $N+1$ samples separated by a fixed time step Δt. The unknowns are the joint configurations and the position along the path,

$$q_0, \ldots, q_N \qquad \text{and} \qquad s_0, \ldots, s_N,$$

so that s_k states where on the path the end-effector is at time $k\,\Delta t$.

Tracking error. At each sample the pose error between the arm and the path is

$$e_k = \begin{bmatrix} e_{p,k} \\ e_{R,k} \end{bmatrix}, \qquad e_{p,k} = p(q_k) - p_d(s_k), \qquad e_{R,k} = \operatorname{Log}\big(R_d(s_k)^T R(q_k)\big)^\vee,$$

where $p(q_k)$ and $R(q_k)$ are the end-effector position and orientation from forward kinematics, and the orientation error uses the logarithmic map on $SO(3)$.

Joint velocity and acceleration. These are approximated by finite differences,

$$\dot{q}_k = \frac{q_k - q_{k-1}}{\Delta t}, \qquad \ddot{q}_k = \frac{q_k - 2q_{k-1} + q_{k-2}}{\Delta t^2}.$$

Cost. The optimization problem is

$$\begin{aligned} \min_{q_0,\dots,q_N,\, s_0,\dots,s_N} \quad & \sum_{k=0}^{N} \|e_k\|_W^2 && \text{(stay on the path)} \\ & + \lambda_v \sum_{k=1}^{N} \|\dot{q}_k\|^2 \;+\; \lambda_a \sum_{k=2}^{N} \|\ddot{q}_k\|^2 && \text{(smooth joint motion)} \\ & + \lambda_s \sum_{k=1}^{N} \big(s_k - s_{k-1} - \tfrac{1}{N}\big)^2 && \text{(keep the requested timing),} \end{aligned}$$

where W weights translational against rotational error, and $1/N$ is the increment corresponding to uniform progress along the path.

Constraints. The trajectory starts from the current configuration and covers the whole path,

$$q_0 = q_{\text{start}}, \qquad s_0 = 0, \qquad s_N = 1,$$

the manipulator limits are imposed componentwise,

$$q_{\min} \le q_k \le q_{\max}, \qquad -\dot{q}_{\max} \le \dot{q}_k \le \dot{q}_{\max}, \qquad -\ddot{q}_{\max} \le \ddot{q}_k \le \ddot{q}_{\max},$$

and the path parameter advances monotonically,

$$0 \;\le\; s_k - s_{k-1} \;\le\; \Delta s_{\max}.$$

The lower bound of zero lets the end-effector pause on the path, which is exactly the behaviour expected at a crossing. The upper bound stops the optimizer from jumping over a difficult section in a few large steps.

Note that the Jacobian is never inverted: it appears only through the derivatives of e_k in the optimization solver. The problem therefore stays well posed at a singular configuration, and a path may be crossed whenever a continuous joint trajectory satisfying the constraints exists.

Two questions from one problem. The same formulation answers both feasibility questions by changing a single setting:

- **Is the path crossable at all?** Take λ_s small and let the timing adapt freely. A solution with acceptable tracking error shows that a continuous joint trajectory exists along the path. Failure indicates a genuinely infeasible path.
- **Is it crossable at the requested speed?** Take λ_s large, or impose $s_k = k/N$ directly, which recovers the usual fixed-time problem under $\dot{q}_{\max}$ and $\ddot{q}_{\max}$.

Sweeping the total motion time $T = N\Delta t$ for a fixed path locates the boundary between the two cases and gives the shortest time in which the crossing can be performed.

Reported quantities.

- position and orientation tracking error, reported separately
- joint velocity and acceleration profiles
- the resulting Cartesian speed, in particular its minimum near the singular configuration

4 Scope and Constraints

- The project is simulation-based using ROS2 and MoveIt2.
- Both exact and near-singularity trajectories should be tested. Near-singular configurations are especially important because modelling errors and real trajectories rarely pass through an exact mathematical singularity.
- Joint position, velocity and acceleration limits must be included in the evaluation.
- The primary goal is a clean comparison and quantitative analysis, not the development of a real-time industrial motion planner.

5 Requirements

We are looking for a motivated student interested in robot manipulation, numerical methods and trajectory optimization.

- Solid understanding of linear algebra, including Jacobians, pseudoinverses and singular values.
- Basic knowledge of robot kinematics and inverse kinematics.
- Familiarity with optimization, or willingness to learn nonlinear optimization during the project.
- Strong programming skills in Python and C++
- Comfortable working on Linux.
- Good analytical mindset and interest in evaluating algorithms experimentally.
- Clear written and verbal communication.
- Able to work both independently and within a student team.

6 Deliverables

- A reproducible simulation and benchmark environment for singularity-crossing Cartesian trajectories.
- Implementations of conventional inverse-kinematics baselines and the proposed trajectory-level method.
- A set of deterministic and randomized singularity-crossing experiments.
- Quantitative results comparing Cartesian position/orientation error, joint velocity and acceleration, crossing success and computational cost.
- A practical analysis of how Cartesian speed, actuator limits and distance from the singularity affect the feasibility and accuracy of the motion.
- A final report covering the mathematical background, implementation, experiments and conclusions.

7 Resources

- Lynch, K. and Park, F., *Modern Robotics*, https://hades.mech.northwestern.edu/index.php/Modern_Robotics
- Nakamura, Y. and Hanafusa, H., "Inverse kinematic solutions with singularity robustness for robot manipulator control," ASME Journal of Dynamic Systems, Measurement, and Control, 1986.
- Chiaverini, S., Siciliano, B. and Egeland, O., "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," IEEE Transactions on Control Systems Technology, 1994.
- Nenchev, D. N., Tsumaki, Y. and Uchiyama, M., "Singularity-consistent parameterization of robot motion and control," International Journal of Robotics Research, 2000.
- Schreiber, G., Otter, M. and Hirzinger, G., "Solving the singularity problem of non-redundant manipulators by constraint optimization," IEEE/RSJ IROS, 1999.
- Pinocchio: https://stack-of-tasks.github.io/pinocchio/
- CasADi: https://web.casadi.org/
- IPOPT: https://coin-or.github.io/Ipopt/
- MoveIt 2 Humble, Robot Model and Robot State: https://moveit.picknik.ai/humble/doc/examples/robot_model_and_robot_state/robot_model_and_robot_state_tutorial.html
- MoveIt 2 Humble, RobotState API (forward kinematics, Jacobians and Cartesian paths): https://moveit.picknik.ai/humble/api/html/classmoveit_1_1core_1_1RobotState.html
- MoveIt 2 Humble, Kinematics Configuration: https://moveit.picknik.ai/humble/doc/examples/kinematics_configuration/kinematics_configuration_tutorial.html
- MoveIt 2 Humble, KinematicsMetrics (singular values, manipulability index and manipulability ellipsoid): https://moveit.picknik.ai/humble/api/html/classkinematics__metrics_1_1KinematicsMetrics.html
- MoveIt 2 Humble, MoveGroupInterface: https://moveit.picknik.ai/humble/doc/examples/move_group_interface/move_group_interface_tutorial.html
- MoveIt 2 Humble, MoveIt Servo: https://moveit.picknik.ai/humble/doc/examples/realtime_servo/realtime_servo_tutorial.html
- MoveIt 2 Humble, Trajectory Processing and Time Parameterization: https://moveit.picknik.ai/humble/doc/concepts/trajectory_processing.html
- ROS 2 Control Humble, JointTrajectoryController: https://control.ros.org/humble/doc/ros2_controllers/joint_trajectory_controller/doc/userdoc.html
- ROS 2 Control Humble, JointTrajectoryController Trajectory Representation and Interpolation: https://control.ros.org/humble/doc/ros2_controllers/joint_trajectory_controller/doc/trajectory.html