Kinematic Path Planning Under Uncertainty

Capprin Bass*, Neha Pusalkar†, and Brett Stoddard‡
Collaborative Institute for Robotics and Intelligent Systems (CoRIS), Oregon State University
Corvallis, Oregon
Email: {*basscap, †pusalkan, ‡stoddabr}@oregonstate.edu

***Abstract*— Recently in the robotics industry, companies construct certain research robots from cheaper components, lowering the barrier of entry to study robot systems. These systems pose a challenge from a planning and control standpoint, as uncertainty in actuation and sensing propagates into the trajectory of the robot. In this paper, we present a novel method of expressing joint space uncertainty in the task space, respecting system kinematics. We cast this expression for uncertainty as a pathlength metric, reflecting uncertainty accrued over a trajectory. Finally, we use our covariant pathlength metric as a heuristic for path planning (Covariant A*), producing paths for a planar manipulator that minimize uncertainty. The resultant algorithm enables finding the path that optimizes for the minimizing end effector uncertainty at the final position.**

I. Introduction

Modern industrial robots are built to be controlled using kinematic and dynamic principles. Many systems take advantage of powerful actuators, precise sensors, and fast computers to make software control tenable, using these mechanical models. Task space path plans can be followed with PID control on top of inverse kinematics or dynamics; the result is a robust stack of methods that has seen widespread use in industry [1]. However, this approach literally comes at a cost: industrial robots range in price from tens to hundreds of thousands of dollars.

A recent trend in robotics is the construction of lower cost systems, often with the goal of a lower barrier to entry in robotics research. Some current examples of robots built with this goal include Hello Robot's Stretch [2], Pollen Robotics' Reachy [3] and others, as well as inexpensive hobby and education-oriented robots [4]. These systems often may be characterized as the antithesis of current industrial robots: inexpensive actuators, sensors, and computers are chosen on purpose to keep cost low. The relative imprecision of constituent components makes control of these systems a challenge. Uncertainty must be built into models for system behavior, and ought to be accounted for when designing motion plans.

In this paper, we address the need for uncertainty-aware path planning with a novel approach, taking advantage of first-order system kinematics to represent uncertainty at the end effector. Unlike many previous approaches [5], where uncertainty is incorporated into control systems, we leverage kinematic uncertainty at the path planner level.

Foundational work in robotics established mapping uncertainty in the configuration or joint space to workspace [6]. Singularity, and collision avoidance can be done by mapping constraints between joint and work spaces. Our approach takes a similar method of mapping between joint space uncertainty and the task space to provide an intuitive representation of uncertainty at any configuration using the Jacobian.

We use the covariance between actuators to represent uncertainty in the joint space and map it up to the task space using an *inverse pullback* operation:

$$M = \left(J^{\dagger}\right)^{T} \Sigma \, J^{\dagger}, \tag{1}$$

where M is the uncertainty expressed in the task space, J is the jacobian of the manipulator, and Σ is the joint-space defined covariance. The task space uncertainty M may be used as a length metric for paths defined in the task space; it reflects the uncertainty accrued by taking a given path. We demonstrate our covariant pathlength metric as a heuristic for path planning, producing paths that respect system uncertainty.

The remainder of this paper is organized as follows. In §II, we review the relevant background in kinematics, geometry, and path planning. In §III, we describe how the covariance matrix is mapped into the task space, and how uncertainty is accounted for in path planning. In §IV, we demonstrate our method on a planar manipulator, generating low-uncertainty paths. In §V, we make final remarks and comment on future work.

II. Background

A. Model and Kinematic Map

When studying robot systems, we usually distinguish between the joint space and the task space. The joint space Θ refers to the set of actuator configurations, and the task space G refers to the set of end effector or baseframe configurations. In this paper, we focus on revolute manipulators in the plane; this implies that $\Theta = S^N$, where N is the number of joints, and $G = SE(2)$.

The robotics community uses a number of methods to move between the joint space and the task space. These are often referred to as forward kinematics, inverse kinematics, and their dynamic counterparts. The most important mapping in the context of this paper is the Jacobian, which maps between tangent spaces. In our case, for $\theta \in \Theta$, this is:

$$J(\theta) : T_{\Theta} \to T_G. \tag{2}$$

The Jacobian may be interpreted as a first order kinematic map; it linearly maps velocity in the joint space at a given

configuration to velocity in task space. For $g \in G$ and $\theta \in \Theta$, this is:

$$\dot{g} = J(\theta)\dot{\theta}. \tag{3}$$

B. Bilinear Forms

Here, the Jacobian is used to map velocity-dependent quantities (represented as bilinear forms) from the joint space to the task space. Bilinear forms are usually defined as symmetric or skew-symmetric matrices Ω:

$$\Omega = \begin{bmatrix} \Omega_{11} & \Omega_{12} & \Omega_{13} & \cdots \\ \pm\Omega_{12} & \Omega_{22} & \Omega_{23} & \cdots \\ \pm\Omega_{13} & \pm\Omega_{23} & \Omega_{33} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{4}$$

and operate on velocities $\dot{\theta}_1, \dot{\theta}_2$ using matrix multiplication:

$$\Omega(\dot{\theta}_1, \dot{\theta}_2) = (\dot{\theta}_1)^T \Omega\, \dot{\theta}_2. \tag{5}$$

The key benefit to this representation is that the Jacobian may be used to relate forms defined on the joint space to quantities in the task space. From (3), we relate task space velocities to joint space velocities:

$$\dot{\theta} = J^\dagger \dot{g}, \tag{6}$$

where $J^\dagger$ refers to the pseudoinverse of the Jacobian, and respects the fact that J is not always invertible. Now, (6) is applied to (5) to *pullback* workspace velocities by the inverse of the kinematic map:

$$\Omega(\dot{g}_1, \dot{g}_2) = \overbrace{(\dot{g}_1)^T \underbrace{(J^\dagger)^T}_{\Omega^G} \Omega^\Theta \overbrace{J^\dagger}^{\dot{\theta}_2} \dot{g}_2}^{(\dot{\theta}_1)^T}. \tag{7}$$

Bilinear forms have a broad set of applications; in this paper, we use them in two specific ways. The first is to define covariance over the joint space, as a first order estimate for uncertainty. The second is to define a Riemannian metric, which measures distance over a space, and is essential for alternate definitions of pathlength for planning.

C. Covariance Matrix

In probability and statistics, variance σ measures the expected (or average) deviation from the mean in a random variable. As an extension, *covariance* Σ between two (or more) random variables measures how each effects the other, i.e. how correlated the two distributions are. As an example, for random variables X and Y, the covariance Σ may be written as a matrix,

$$\Sigma = \begin{bmatrix} \sigma_{XX} & \sigma_{XY} \\ \sigma_{YX} & \sigma_{YY} \end{bmatrix}, \tag{8}$$

where the diagonal σ_{ii} terms refer to the variance of each distribution, and the off-diagonal σ_{ij} terms refer to the contribution of the *other* distribution towards variance.

Due to the shared properties[1] between covariance and general bilinear forms, we may interpret covarariance as a bilinear form. When we do so, *vectors* are linear combinations of random variables, and matrix multiplication yields the variance of said linear combination. For a linear combination vector c, the variance $\sigma(c)$ is

$$\sigma(c) = c^T \Sigma\, c. \tag{9}$$

We now describe Riemannian metrics, a related application of bilinear forms, generally defining distance.

D. Riemannian Metrics

Formally, a Riemannian metric defines an inner product on the tangent space (here, T_Θ or T_G). The inner product can be thought of as a generalization of the dot product: it measures (by some choice of measurement) the similarity of two vectors. A useful consequence of the inner product is that a vector may be measured against *itself* to provide a measure of length.

On a broader scale, lengths are measured by integrating the metric (represented as a bilinear form) over a path definition. For a metric M, path parametrization $\phi(t)$, and location $x(t)$, the weighted length L_ϕ is

$$L_\phi = \int_\phi \left((\dot{x})^T M(x)\, \dot{x}\right)^{\frac{1}{2}} d\phi. \tag{10}$$

In the discrete case, weighted length of a path with n segments is given by

$$L_\phi = \sum_i^{n-1} \left(\Delta x_{i,i+1}^T M(x_i) \Delta x_{i,i+1}\right)^{\frac{1}{2}}, \tag{11}$$

as we weight the length of each segment $\Delta x_{i,i+1}$ by the metric M.

In this paper, a Riemannian metric is used to define a custom measure of distance. Metric distance applied to a path planning algorithm can produce optimal paths with respect to our own choice of distance.

E. Path Planning

Path planning is a rich area of study, most of which is outside the scope of this paper. However, most planners can take advantage of a number of distance measures as heuristics to compare candidate paths with some formulations being situationally more effective than others [7]; classic examples include p-norms such as Euclidean or Manhattan distance. Here, we use a simple path planner, A* search, to find shortest paths under metric distance.

A* search is a heuristic based path planning algorithm. Each node in the A* graph stores the cost of the cheapest path from start to the current node (gScore) and an estimate of the cost to reach the goal node from current node (hScore). The node also stores the best guess (fScore) of the path cost, if the final path were to use the current node, which is the sum of gScore and hScore. A list of frontier nodes and explored nodes is maintained to decide which nodes to choose for further expansion. The node with the lowest fScore is expanded by selecting the neighbouring nodes using 8-connectivity. If the neighbour node was already in the frontier nodes list, its gScore and fScore is updated, else this neighbouring node is added to the frontier nodes list. This process continues until the goal node is reached.

[1]Specifically, the important shared properties between covariance and bilinear forms are biliniarity, symmetry, and positive semi-definiteness.

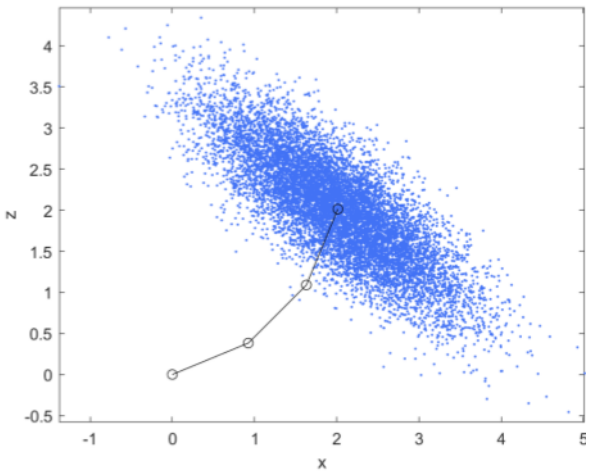


Fig. 1: Joint space multiviariate normal distribution, pulled back through first order system kinematics. Resulting work space samples lie in an ellipse, centered about the end effector.

III. Covariance as a Pathlength Metric

Here, we draw on the methods introduced in §II to represent uncertainty respecting system kinematics. We briefly express our method using the relevant mathematics, and explain algorithmic implementation. We integrate our approach with path planning, generating uncertainty-aware motion plans.

A. Kinematic Mapping of the Covariance Matrix

The key innovation of this work is casting covariance as a pathlength metric. We define uncertainty in the joints of a system using a covariance matrix defined on the joint space. The variance σ generated by a set of joint motions $\dot{\theta}$ is given by the covariance matrix Σ as

$$\sigma(\dot{\theta}) = (\dot{\theta})^T \Sigma\, \dot{\theta}. \tag{12}$$

By interpreting covariance Σ as a bilinear form, we can represent it on the task space. We map Σ from the joint tangent space T_Θ to the task tangent space T_G using an inverse pullback:

$$\sigma(\dot{g}) = \overbrace{(\dot{g})^T}^{(\dot{\theta})^T} \underbrace{(J^\dagger)^T}_{M}\, \Sigma\, \overbrace{J^\dagger}^{\dot{\theta}}\, \dot{g} \longrightarrow M(\theta) = (J^\dagger)^T \Sigma\, J^\dagger, \tag{13}$$

which can be written simply as

$$\sigma(\dot{g}) = (\dot{g})^T M\, \dot{g}. \tag{14}$$

The task space covariance $M(\theta)$ can be interpreted in a similar way to a manipulability ellipsoid. For any configuration θ, covariance $M(\theta)$ can be drawn as an ellipse at the end effector of the manipulator, where the major axis encodes the direction generating the most variance. Fig. 1 demonstrates this empirically, for joint samples drawn from the multivariate normal distribution, pulled back through the system kinematics.

Algorithm 1: Computation of Metric Pathlengths

Input: Arm geometry $geom$, Path $path$, Variance Σ
Output: Pathlength len
$len \leftarrow 0$
for *lastPoint, point, nextPoint* **in** *path* **do**
  `//use Newton-Rhapson for arm shape`
  $shape \leftarrow$ inverseKinematics$(lastPoint, point)$
  $J \leftarrow$ bodyJacobian$(geom, shape)$
  `//compute metric`
  $M \leftarrow (J^\dagger)^T \Sigma J^\dagger$
  `//measure segment vector`
  $g = (nextPoint - point)$
  $len \leftarrow len + \left((g)^T M\, g\right)^{\frac{1}{2}}$
end
return *len*

We now interpret covariance as a Riemannian metric on the task space. As in (10), the aggregate variance (or, uncertainty) σ_ϕ generated by a path $\phi(t)$ in the task space can now be written as

$$\sigma_\phi = \int_\phi \left((\dot{g})^T M(\theta(t))\, \dot{g}\right)^{\frac{1}{2}} d\phi. \tag{15}$$

We use this measure for length to perform path planning in the task space, selecting paths that minimize accrued uncertainty.

B. Planning Under Uncertainty

For planning to take advantage of the covariant metric, we require the configuration of the manipulator at each point along the path. We then compute lengths using (15), used both to compute path and heuristic lengths. This process is described here, as well as in Algorithm 1.

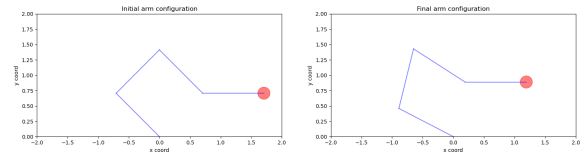
We define a path that starts at the initial position of the end effector of a manipulator. For each subsequent point on the path, we perform Newton-Rhapson inverse kinematics to find valid configurations generating the path (a classic inverse kinematics problem). Given the configuration of the arm, we compute the body Jacobian of the end effector at each point.

With the Jacobian of the manipulator, we can compute the task space covariance at each point using (13). This is then used to find weighted lengths of each segment along the path. The total covariance-weighted length of the path is then a sum of each weighted segment, as in (11).

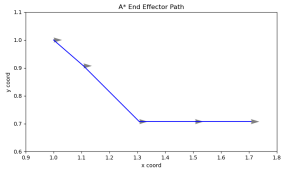
Weighted path lengths are used in planning as an alternative to Euclidean (or other) distance measures. Here, as a simplified example, we use covariant metric length with A* search to generate approximately minimum-uncertainty paths. We now demonstrate this approach, and compare the paths generated for Euclidean and covariant distance measures.

IV. Results and Analysis

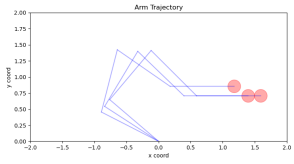
We demonstrated the use of path planning using our uncertainty metric using different arms in SE(2). Paths for the arms in SE(2) were calculated for both a standard A* path



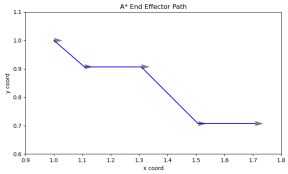
(a) Initial and final arm positions



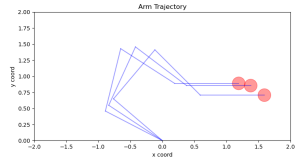
(b) EE Path for Covar. A*



(c) Trajectory for Covar. A*

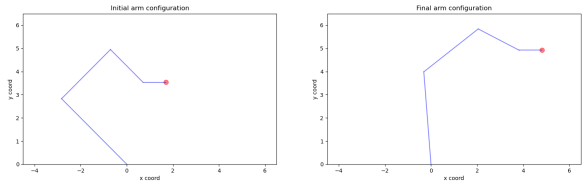


(d) EE Path for Euclid. A*

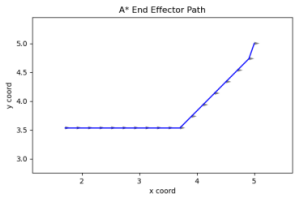


(e) Trajectory for Euclid. A*

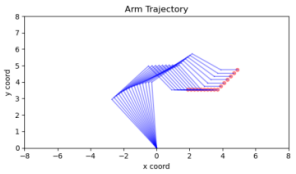
Fig. 2: Resultant paths for an arm 1 which has links of the identical sizes. Middle row: results using the covariance path planner. Bottom row: results using the euclidean path planner.



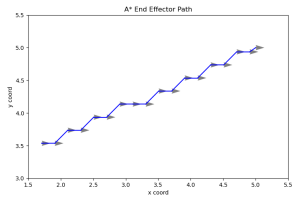
(a) Initial and final arm positions



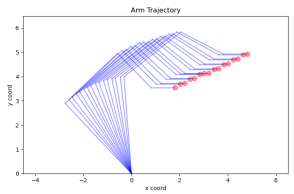
(b) EE Path for Covar. A*



(c) Trajectory for Covar. A*



(d) EE Path for Euclid. A*



(e) Trajectory for Euclid. A*

Fig. 3: Resultant paths for an arm 2 which has links of the varying sizes. Middle row: results using the covariance path planner. Bottom row: results using the euclidean path planner.

planner (i.e., minimizing the path length in task space), and our A* with the covariant metric as a cost (i.e., minimizing the end effector uncertainty along the path). The results of our path planner[2], Covariant A*, compared to traditional Euclidean A* are shown in Figures 2 and 3.

To enable the A* path planning algorithm, the task space of the end effector was discritized into an 8-connected grid world. Each node in this grid world consisted of an x, y, and θ coordinates. Adjacent nodes in this grid world included nine x, y positions by three angles θs (+90deg, 0deg, -90deg) which results in each node in the grid world graph having up to 27 neighbors. Only nodes which had a valid IK solution were explored. This essentially captures holonomic motion of the end effector in the task space.

Paths for two different robot arms are presented. These two arms are over-actuated, consisting of four links each, as our hueristic is incalculable when a node is explored outside of the robot arm's workspace. Resultant paths for our Covariant A* algorithm are shown appear to result in a smoother path with fewer angle changes vs Euclidean distance.

V. Conclusion

We present a path planning formulation based on minimizing the final uncertainty of an end effector in task space. This is based on a novel formulation of a Riemannian metric from joint covariance and the Jacobian. We apply this metric as a heuristic for an A* path planner to generate valid trajectories that minimizes propagated uncertainty. We show results of using this path planner on two robot arms in SE(2) compared to a traditional path planner (A* with Euclidean distance heuristic)

Future work can expand our implementation to SE(3) and improve performance by using other path planning algorithms. For example, incorporating our heuristic within Lazy Receding-Horizon A* [8] may greatly improve the time to find an optimal path as calculating the full covariance cost for a node is computationally expensive. Additionally, Monte Carlo simulations showing propagating uncertainty in joint space across a path in task space could be used to better quantify the benefit of our approach.

Optimizing path plans based on kinematic uncertainty, as we have done here, will enable more relaxed control systems and therefore more inexpensive robots.

VI. Member Contributions

Capprin: Problem formulation and calculation of the uncertainty heuristic.
Neha: Implementation of A* path planning with traditional Euclidean distance heuristic and the covariant metric heuristic.
Brett: Synthesis of results and visualizations.

References

[1] R. P. Borase, D. Maghade, S. Sondkar, and S. Pawar, "A review of pid control, tuning methods and applications," *International Journal of Dynamics and Control*, vol. 9, no. 2, pp. 818–827, 2021.

[2]Code available at `https://github.com/NehaPusalkar/Rob545_project`

[2] C. C. Kemp, A. Edsinger, H. M. Clever, and B. Matulevich, "The design of stretch: A compact, lightweight mobile manipulator for indoor human environments," *arXiv preprint arXiv:2109.10892*, 2021.

[3] S. Mick, M. Lapeyre, P. Rouanet, C. Halgand, J. Benois-Pineau, F. Paclet, D. Cattaert, P.-Y. Oudeyer, and A. De Rugy, "Reachy, a 3d-printed human-like robotic arm as a testbed for human-robot control strategies," *Frontiers in neurorobotics*, p. 65, 2019.

[4] Z. Lu, A. Chauhan, F. Silva, and L. S. Lopes, "A brief survey of commercial robotic arms for research on manipulation," in *2012 IEEE Symposium on Robotics and Applications (ISRA)*. IEEE, 2012, pp. 986–991.

[5] B. Xiao, L. Cao, S. Xu, and L. Liu, "Robust tracking control of robot manipulators with actuator faults and joint velocity measurement uncertainty," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 3, pp. 1354–1365, 2020.

[6] V. Lumelsky and K. Sun, "A unified methodology for motion planning with uncertainty for 2d and 3d two-link robot arm manipulators," *The International journal of robotics research*, vol. 9, no. 5, pp. 89–104, 1990.

[7] X. Liu and D. Gong, "A comparative study of a-star algorithms for search and rescue in perfect maze," in *2011 international conference on electric information and control engineering*. IEEE, 2011, pp. 24–27.

[8] A. Mandalika, O. Salzman, and S. Srinivasa, "Lazy receding horizon a* for efficient path planning in graphs with expensive-to-evaluate edges," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 28, 2018, pp. 476–484.