Direct Self-Identification of Inverse Jacobians for Dexterous Manipulation Through Particle Filtering

Joshua T. Grace[1], Podshara Chanrungmaneekul[2], Kaiyu Hang[2], and Aaron M. Dollar[1]

***Abstract*— The ability to plan and control robotic in-hand manipulation is challenged by several issues, including the required amount of prior knowledge of the system and the sophisticated physics that varies across different robot hands or even grasp instances. One of the most direct models of in-hand manipulation is the inverse Jacobian, which can directly map from the desired in-hand object motions to the required hand actuator controls. However, acquiring such inverse Jacobians without complex hand-object system models is typically infeasible. We present a method for controlling in-hand manipulation using inverse Jacobians that are self-identified by a particle filter-based estimation scheme that leverages the ability of underactuated hands to maintain a passively stable grasp during self-identification movements. This method requires no *a priori* knowledge of the specific hand-object system and learns the system's inverse Jacobian through small exploratory motions. Our system approximates the underlying inverse Jacobian closely, which can be used to perform manipulation tasks across a range of objects successfully. With extensive experiments on a Yale Model O hand, we show that the proposed system can provide accurate in-hand manipulation of sub-millimeter precision and that the inverse Jacobian-based controller can support real-time manipulation control of up to 900Hz.**

I. Introduction

Enabling robots to perform similar manipulation tasks to human hands has been a continuing challenge in robotics. Dexterous manipulation is defined as the capability to reposition or reorient an object within a grasp [1] and is perhaps more appropriately referred to as in-hand or within-hand manipulation. In-hand manipulation is a vital skill that humans use for handwriting, placing objects, and unscrewing bottle caps, making it a prerequisite for robots to interact with objects designed for humans. A difficulty in dexterous, in-hand manipulation is the variability in hand-object systems, which refers to the system controlled by the actuators in the fingers and linked through the grasp to an object. This variation requires adapting the controller to the specific hand-object system.

Manipulation of hand-object systems has been performed using analytical system models. These approaches include modeling the curvature at the contact point [2], the finger contacts [3], the kinematic model of the hand [4], and the model of the complete hand-object system [5]. While these methods accurately represent specific hand-object systems, they're less applicable to general ones. Additionally, the required model parameters are difficult to determine outside simulation. Our approach requires no *a priori* knowledge of the system and estimates the inverse Jacobian without an explicit model of the system.

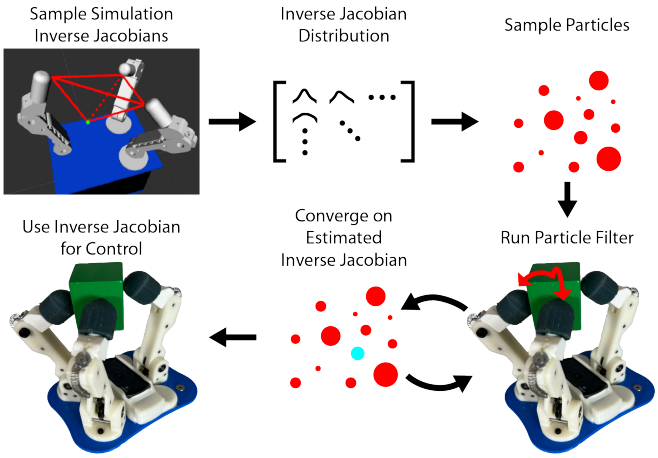


Fig. 1: Overview of inverse Jacobian-based particle filter

To overcome the reliance on accurate models to perform manipulation, deep learning models have been applied to learn in-hand manipulation [6]–[8]. These methods require large datasets to be trained successfully, considerably increasing offline computational requirements. Additionally, learning-based methods usually don't generalize well to cases outside the training data. So, they're challenging to use in the real world. Our method uses a distribution determined numerically using the set of possible joint locations of the fingers. So, it can generalize to a wide range of objects without reliance on specific object morphologies.

We consider manipulation using underactuated hands, which are highly capable of grasping and manipulation as their ability to passively adapt to grasped objects enables them to maintain stable grasps over a large set of hand movements [19]–[22]. However, the low number of controlled degrees of freedom makes predicting task space movements difficult. While these problems can be solved with extensive sensing on the hand, we aim to enable manipulation with limited sensing capabilities to avoid adding hardware requirements. We leverage the stability of underactuated hands to limit finger slip and use a single camera to track object movements to adapt our controller to the hand-object system.

Several manipulation approaches have been demonstrated for underactuated systems. In [9], [10], Jacobians are used to simplify the complex motions of deformable objects. While

[1]Department of Mechanical Engineering & Materials Science, Yale University, New Haven, CT 06520 USA josh.grace@yale.edu; aaron.dollar@yale.edu

[2]Department of Computer Science, Rice University, Houston, TX 77005, USA. pc45@rice.edu; kaiyu.hang@rice.edu

This work was supported by the US National Science Foundation grants FRR-2132823 and FRR-2133110 and funded in part by the Boston Dynamics AI Institute.

TABLE I: Comparison of relevant manipulation approaches

Manipulation Algorithm	**Manipulation Type**	**Limitation**	***A Priori* Knowledge**
Deformation Jacobian [9]	Deformable Object	Requires manipulator kinematic knowledge/ controller	System kinematics
Diminishing Rigidity Jacobian [10]	Deformable Object	Requires manipulator kinematic knowledge/ controller	System kinematics
Precision Manipulation Primitives [11]	2D Precision	Needs hand specific manipulation primitives	Manipulation Primitives
Learned State Transition Model [12]	2D Precision	Intensive model training required for useful controller	Dataset of state, input, and resulting movement for model training
Non-Parametric Self-Identification [13]	2D Precision	Only for 2D manipulation	None
Object-Agnostic Manipulation Model [14]	3D Precision	Intensive model training required for useful controller	Dataset of movement, and resulting pose changes for model training
Learned Control Mappings [15]	3D Precision	Intensive model training required for useful controller	Dataset of movement, and resulting pose changes for model training
Jacobian-Based Deformation Control [16]	3D Manipulation	Manual demonstrations of control target required	None
VLR Particle Filter [17], [18]	3D Precision	Needs processing intensive model of system	Full hand model/ kinematics of hand
Inverse Jacobian Particle Filter	**3D Precision**	**Requires parameter distribution to initialize filter**	**Characterization of inverse Jacobian parameters**

they require full kinematic knowledge of the manipulator, our approach, which does not, is analogous in using Jacobians online to approximate complex manipulation systems. In [11]–[13], planar manipulation with underactuated hands is demonstrated. Our method performs 3D manipulation, which is much more complex, requiring a more capable controller. In [14], [15], 3D manipulation is performed on underactuated hands using learned models. Creating these models requires significant offline training, compared with our method, which only requires a distribution of possible inverse Jacobian parameters. In [16], 3D manipulation is performed with a self-identified Jacobian using no *a priori* knowledge. Unlike our method, it requires operator demonstration of tasks and needs significant sensing on the fingers of the hand. Table I summarizes these related manipulation approaches.

Our previous work [17], [18] is most similar to this approach. It used particle filters to estimate the hand-object system parameters, including joint configurations, link lengths, and spring stiffness, to build a model of the system referred to as the virtual linkage-based representation (VLR). Like this work, using self-identification (self-id), random system inputs and their corresponding outputs are captured and used to estimate the configuration parameters. We improve our previous work by removing the dependence on a model. We estimate the inverse Jacobian, the most direct method of controlling the system, without modeling the underlying system.

A precise system model is not always required to control a robotic system, as long as a closed-loop control approach is applied to the system, an approximate model is often sufficient for achieving control of the system [23]. So, approximate Jacobians are often sufficient to control a system. However, Jacobians can be highly non-linear, particularly for underactuated hands where the space of valid Jacobians is challenging to model analytically due to unpredictable movements in task space. Thus, estimation systems for underactuated hands must approximate the true system Jacobian accounting for these challenges.

We present an approach that estimates a functional inverse Jacobian using particle filters. The process hypothesizes a range of potential inverse Jacobians, executes random movements of the hand actuators, monitors the motion of the target point on the grasped object, and resamples inverse Jacobian particles that are a poor match for the hidden state, iterating until the filter converges to an estimated inverse Jacobian. As a result, we can control a hand-object system without *a priori* knowledge of the system itself.

Our results show that avoiding a model makes our method much more efficient than our previous approach, running at 74.9Hz compared with ~ 0.1666Hz for the VLR particle filter, as it doesn't need heavy processing to use a model. Furthermore, the new method can estimate a usable controller with fewer self-id iterations, reducing the required filter iterations from 15.1 to 5.4. These improvements are achieved without a significant reduction in control accuracy.

II. Problem Formulation

We consider the problem of manipulation using robotic hands with limited finger slip. When the fingertips of the hand grasp an object, the object's pose becomes controllable using the finger actuators. We're particularly interested in controlling a point on the manipulated object we call the Point of Manipulation (POM), χ, which is rigidly attached to the object and kinematically linked to the finger actuators of the hand through contact points on the fingertips [17].

We perform manipulation using the Jacobian $J(\theta)$, which linearizes the map from finger joint angle rates $\dot{\theta}$ to the spatial twist V in SE(3) [23]. The Jacobian which maps to the spatial twist V_f in frame f, $J_f(\theta)$ is given by

$$V_f = J_f(\theta)\dot{\theta} \tag{1}$$

To transform the Jacobian reference frame from f to the POM χ, the adjoint representation $[Ad_T]$ of the transform T from frame f to χ is applied to $J_f(\theta)$ to calculate $J_\chi(\theta)$

$$V_\chi = [Ad_T]J_f(\theta)\dot{\theta} = J_\chi(\theta)\dot{\theta} \tag{2}$$

The inverse of this equation maps spatial twist to joint rates

$$\dot{\theta} = J_\chi(\theta)^{-1}V_\chi \tag{3}$$

As the Jacobian isn't guaranteed to be square, the inverse Jacobian $J_\chi(\theta)^{-1}$ is computed with the Moore-Penrose Psuedoinverse $J_\chi(\theta)^+$. Traditionally, $J_\chi(\theta)^+$ would be calculated in a closed-form manner using the system kinematics and a known transformation from f to χ to calculate $J_\chi(\theta)$, and inverting the matrix to obtain $J_\chi(\theta)^+$. But, we avoid requiring finger joint sensing or prior information about the manipulated object to ensure generalizability, so there is insufficient knowledge of the system to compute the inverse Jacobian directly. Thus, the inverse Jacobian at time t, $J_\chi(\theta_t)^+ = J^+$ must be approximated by $\hat{J}^+$.

Additionally, approximating the inverse Jacobian handles the unpredictability of underactuated mechanisms. There are infinite combinations of proximal and distal joint angles for a motor configuration, and the precise configuration can't be known *a priori*. But, for a given grasp, the joints move to minimize the energy in the system, which is repeatable and enables the approximation of the underlying kinematics [24].

We develop a function Γ which estimates an inverse Jacobian $\hat{J}^+$ given random samples $x_1, x_2, \ldots, x_n$ of the joint rates and spatial twist.

$$\Gamma : (x_1, x_2, \ldots, x_n) \mapsto \hat{J}^+ \tag{4}$$

Assuming the estimated inverse Jacobian $\hat{J}^+$ produced by Γ approximates J^+, we can then control the manipulation system by predicting the required joint rates $\dot{\theta}$ for a desired spatial twist V_s as in Equation (3).

III. Estimating Inverse Jacobians Using Particle Filters

A. Particle Filter

We use a particle filter to implement Γ in Equation (4). Particle filters are a class of non-parametric filters shown in [17], [18], [25] to successfully estimate hidden states for hand-object systems. The filter creates a series of hypothesized system states and tests how well each hypothesis matches the true system, allowing it to handle highly non-linear systems [26]. Particle filters do not need a complete model of the underlying system, and instead observe how well each particle's output prediction matches the true value. As particle filters can handle the challenge of modeling underactuated systems and the non-linearity of Jacobian spaces effectively, they are an attractive approach for estimating inverse Jacobians for manipulation.

Algorithm 1 Inverse Jacobian Particle Filter

1: $\boldsymbol{\rho} \leftarrow$ GenerateInitialParticles($distribution, N$)
2: **while** !Converged($\boldsymbol{\rho}$) **do**
3: $\quad \chi_{t-1} \leftarrow$ SamplePOM($\cdot$)
4: $\quad u_t \leftarrow$ GenerateRandomMovements($\cdot$)
5: $\quad$ MoveHand(u_t)
6: $\quad \chi_t \leftarrow$ SamplePOM($\cdot$)
7: $\quad u_{pred} \leftarrow$ Predict($\boldsymbol{\rho}, \chi_{t-1}, \chi_t$)
8: $\quad w \leftarrow$ Normalize(ComputeWeight($\boldsymbol{\rho}, u_{pred}, u_t$))
9: $\quad \boldsymbol{\rho} \leftarrow$ Resample($\boldsymbol{\rho}, w$)
10: **end while**
11: **return** Avg($\boldsymbol{\rho}, w$)

The filter uses range of N particles, $\boldsymbol{\rho}_t = [\rho_t^1, \rho_t^2, ..., \rho_t^N]$, which represent the possible system parameters. The probability associated with a given particle ρ_t^i is computed via

$$\rho_t^i \sim p(\rho_t | z_{1:t}, u_{1:t}) \tag{5}$$

where $u_{1:t}$ are the self-id inputs, and $z_{1:t}$ are the observed change in POM pose for those inputs [17]. Assuming that the space of inverse Jacobians is sampled accurately by $\boldsymbol{\rho}$, then Equation (5) should represent the underlying distribution of inverse Jacobians. Unfortunately, this rarely occurs in practice, and the particle filter instead uses a proposal distribution π to iteratively approximate the distribution with

$$\pi(\boldsymbol{\rho}_t) \sim p(\boldsymbol{\rho}_t | \rho_{t-1}, u_t) \pi(\boldsymbol{\rho}_{t-1}) \tag{6}$$

If the particles $\boldsymbol{\rho}$ are sampled from π, using importance sampling, π and p are related by

$$\int_{\boldsymbol{J}^+} p(\boldsymbol{\rho}_t) d\boldsymbol{\rho}_t = \frac{1}{\sum_{i=1}^N w_t^i} \sum_{i=1}^N I(\rho_t^i \in \boldsymbol{J}^+) w_t^i, \quad \rho_t^i \in \boldsymbol{\rho}_t \tag{7}$$

$$w_t^i = exp(-\frac{\|u_t - \rho_t^i \chi_\Delta\|}{2\sigma^2}) \tag{8}$$

where $\boldsymbol{J}^+$ is the space of valid inverse Jacobians for the system, w_t^i is the weight of ρ_t^i computed in Equation (8) using a Gaussian radial bias function [27]. Weights are based on the similarity of the movement predicted by ρ_t^i for the observed change in POM pose change χ_Δ to the true movement u_t. The summation is normalized by the sum of the weights to account for differences between π and p to ensure the weighted $\boldsymbol{\rho}_t$ approximates the probability. Particles are then resampled according to their weights using stratified resampling as described in [26]. Each resampled particle is moved by a random amount along each of its parameter axes using the weighted standard deviation of the particles. This is implemented in lines 7-9 of Algorithm 1. The process is repeated until the particles in the filter converge to an estimated inverse Jacobian $\hat{J}^+$, determined by the norm of the element-wise standard deviation of the particles

$$\|\sigma(\rho_{k_{[i,j]}})\| < \varepsilon \quad \forall k \in [t-2, t-1, t] \tag{9}$$

where σ computes the standard deviation and $\rho_{k_{[i,j]}}$ is the i, j element of the particles at time k. Equation (9) ensures that the particles have remained close together for three filter iterations. When this occurs, the particles have a similar hypothesis for the inverse Jacobian of the system, and there is limited improvement from continuing to run the filter. So, the filter has completed and returns the weighted average of the particles as described in Algorithm 1. The estimated inverse Jacobian can be used to control the hand-object system using Algorithm 2.

Algorithm 2 Manipulation Using Inverse Jacobian

1: **for all** $\chi_{t+1} \in$ waypoints **do**
2: $\quad$ **while** Norm($\chi_{t+1} -$ SamplePOM($\cdot$)) $> \varepsilon$ **do**
3: $\quad\quad \chi_t \leftarrow$ SamplePOM($\cdot$)
4: $\quad\quad \chi_\Delta \leftarrow \chi_{t+1} - \chi_t$
5: $\quad\quad u_t \leftarrow \hat{J}^+ * \chi_\Delta$
6: $\quad\quad$ MoveHand(u_t)
7: $\quad$ **end while**
8: **end for**

B. Estimating Inverse Jacobians

The particle filter is used to estimate an inverse Jacobian $\hat{J}^+$, approximating the true inverse Jacobian J^+ of the system. At the start of the filter, the range of possible inverse Jacobians is sampled. During the initialization phase of the filter, each actuator of the system is moved a random amount

u_t, which moves the POM, giving the particle filter insight into the system. The POM pose χ_{t-1} is sampled before, and χ_t is sampled after the movement of the fingers to find χ_Δ. This is achieved by subtracting the position component of χ_{t-1} from χ_t in Cartesian space to find the difference in position and multiplying the quaternion of χ_{t-1} by the conjugate of the quaternion of χ_t to find rotational change.

As shown by [28], Jacobians can be used for control in discrete time. To use $\hat{J}^+$ in discrete time, we treat the joint rates and spatial twist in Equation (3) as the change in motor position and change in POM pose, χ_Δ, between control steps to produce the equation used for POM control.

$$u_t = \hat{J}^+ \chi_\Delta \tag{10}$$

At each step, we compute the required control input u_t for χ_Δ using $\hat{J}^+$. Calculations using Jacobians are implemented with efficient linear algebra techniques, so using Jacobians ensures that the filter runs at high frequencies.

Jacobian matrices for the hand-object systems tested in this paper were frequently ill-conditioned. So, estimating a Jacobian matrix and then inverting it would produce vastly different inverse Jacobians, which may be unusable for manipulation. Accordingly, we estimate the inverse Jacobian directly to ensure stability.

C. Generating Inverse Jacobian Distribution

To ensure that the initialization of the particle filter captures the possible inverse Jacobian parameters for the hand, a distribution of the feasible inverse Jacobian parameters is computed offline. This information is captured once per hand design, as it is dependent only on the kinematic parameters of the hand. So, once the offline distribution is computed for a given hand, it captures the possible hand-object configurations for the hand. The filter requires no modifications to control different objects.

We simulated the hand using a simulator shown in [24] to accurately simulate the movements of a physical Model O hand. For each joint on the hand's fingers, an angle θ is uniformly sampled from the possible angles for the joint, and a POM position χ is uniformly sampled within the view frame of the in-palm camera. The simulation keeps the initial distances between the fingertips constant during simulation. This process effectively generates a hand-object system by randomizing the POM pose and fingertip positions.

As visualized in Fig. 2, each simulated finger actuator on the hand is moved forward and backward by 0.00025 radians. Next, the partial derivative of the POM pose change from each finger actuator movement is computed using $\delta_{finger} = \frac{(\chi_t - \chi_{t-1})}{u_t}$ and treated as the actuator's column in the Jacobian. Once the Jacobian columns have been calculated for each actuator, the pseudoinverse of the Jacobian is calculated and added to the distribution. This process is repeated 500,000 times to generate a representative distribution.

The mean and standard deviation of each inverse Jacobian parameter are computed and saved for sampling during the initialization phase of the particle filter. When sampling inverse Jacobians in the particle filter, each parameter is sampled independently and added to a candidate particle. This ensures that grasps not directly sampled in the simulation but close to other configurations in the inverse Jacobian space can still be sampled in the particle distribution.

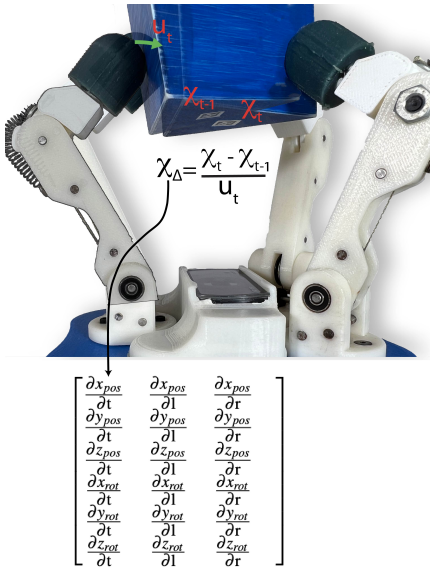


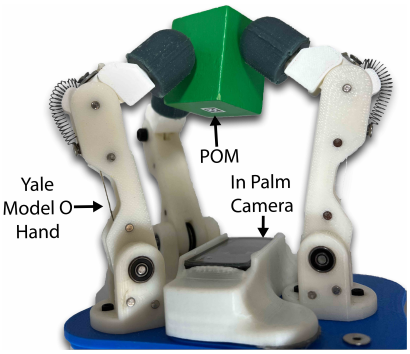
Fig. 2: Sampling method for Jacobian columns. Finger actuation u_t is determined randomly for each finger. Change in the POM pose χ_Δ before and after finger movement is divided by the actuation amount to determine the partial derivative of POM movement

IV. Results

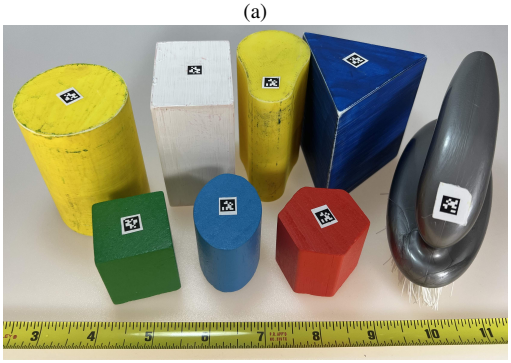
We implement our method on a Yale Model O hand, a three-finger passively adaptive underactuated hand [21]. Each finger has a single actuator driving two encoderless joints. It also uses a 4th actuator to control the abduction angle between the left and right finger, which is set to 45° to maximize the grasping ability of the hand. The hand has a Logitech C920 RGB Camera in the palm to track the POM. The setup is shown in Fig. 3a. A range of object shapes and sizes were used to demonstrate the filter's ability to handle a variety of contact locations and POM positions. Each object's POM is marked with an AprilTag [29] for tracking during experiments. The objects are shown in Fig. 3b.

All code was implemented in Python and uses ROS [30]. The particle filter was run with 10,000 particles sampled from the same precomputed distribution of inverse Jacobians. The filter begins by estimating an inverse Jacobian, as shown in Algorithm 1. An ε of 9×10^{-15} for three consecutive iterations was found experimentally to be an effective convergence threshold. As described in Algorithm 2, the system then navigates between waypoints representing the desired path the POM should move through to perform POM-based manipulation. To move the POM from its current position to the desired waypoint, the required movement of each actuator is calculated using Equation (10). This process is repeated until the norm of the error between the POM and waypoint positions is less than 1mm.

Writing experiments were performed across a range of object shapes, sizes, and POM locations to demonstrate that the inverse Jacobian particle filter can successfully self-



(a)



(b)

Fig. 3: Physical Experiment Setup; 3a: System used for experiments. The hand is a Yale Model O hand, which has been modified with a camera in the palm that tracks the POM on the object being manipulated; 3b Objects used in experiments: From left to right, front to back: Green Cube, Blue Oval, Red Hexagon, Brush, Yellow Cylinder, White Cuboid, Yellow Pear, and Blue Prism

identify and control hand-object systems. These experiments show that our method generalizes to a range of possible hand-object configurations and is robust to real-world noise. Additionally, as both the runtime and convergence time of the filter impact how the system can be used in the real-world, they are also evaluated.

A. *Particle Filter Convergence*

Ensuring the number of iterations required for the particle filter to converge remains low is vital. The self-id process adds time to manipulation tasks because each iteration requires an exploratory motion and should be minimized. All future metrics are reported as mean $\pm$ standard deviation. The inverse Jacobian particle filter converged in 5.43 ± 2.77 iterations during experiments. The particle filter presented in [18] requires 15.1 ± 3.6 iterations to converge to a useful model. So, the method introduced here requires significantly fewer exploratory motions than our previous approaches, demonstrating one of the critical advantages of our model-free method.

B. *Writing Test*

A series of handwriting tasks were performed to evaluate the system. Both the visual path and quantitative error between the goal and actual POM position were analyzed to demonstrate the inverse Jacobian performing manipulation tasks. The writing test figures show that the inverse Jacobian

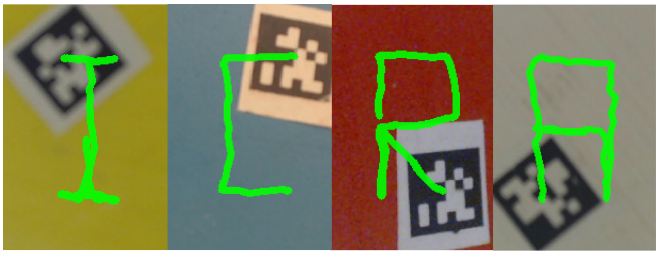


Fig. 4: "ICRA" written using estimated inverse Jacobian with (from left to right) the Yellow Cylinder, Blue Oval, Red Hexagon, and White Cuboid

can move the POM through a trajectory accurately enough to match the goal path visually. The green path is the actual path the POM traveled through during the experiment, while the red path (when present) is the goal path.

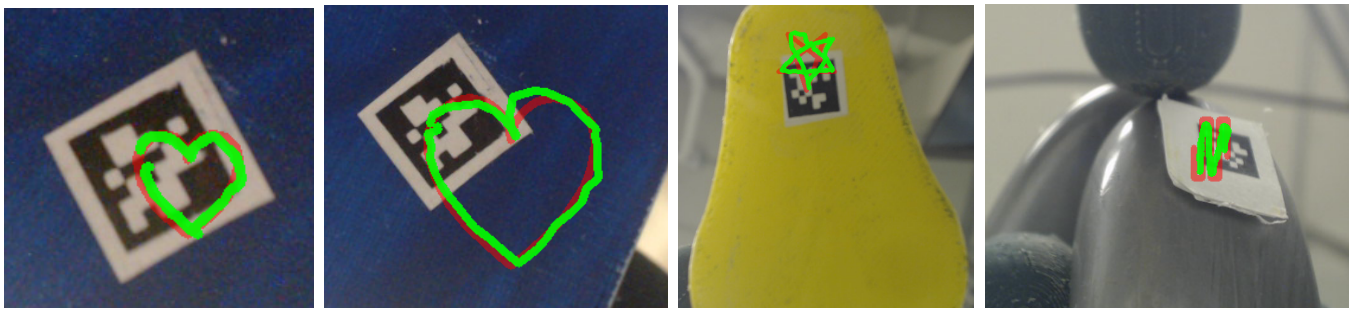
Fig. 4 shows the estimated inverse Jacobian used to write "ICRA" using four different objects. The "A" and "I" were written by larger objects. The white cuboid was 50 x 38 x 75mm tall. The yellow cylinder was 50 x 50 x 77mm. The "C" and "R" were written by smaller objects. The blue oval was 32 x 45 x 40mm. The red hexagon was 37 x 42 x 40mm. These objects demonstrate that the particle filter can successfully manipulate various object shapes, sizes, and grasp contact points.

Fig. 5 shows the estimated inverse Jacobian used to manipulate several objects to draw different shapes. Fig. 5a and 5b use the same object to draw the same path with different sizes. The path shown in Fig. 5a is 5.4 x 5.2mm, while the path in Fig. 5b is 12.1 x 12.4mm. These paths demonstrate large and small-scale movement control. Fig. 5c shows control of a POM away from the object's center. The pear shape used in this experiment is 60 x 38mm wide. The center of the AprilTag is 17mm from the center of the pear on the long axis. This experiment shows control of a POM outside the object's center. Fig. 5d demonstrates the manipulation of a brush with a complicated design, showing the system's application to real-world objects.

These experiments show that the system can accurately control the POM to follow a goal path visually. This shows that the estimated inverse Jacobian is close enough to the true system Jacobian for manipulation tasks across various objects and POMs.

C. *Writing Test Stepwise Error*

The position error of the writing experiments was evaluated at each step to give a quantitative measure of 3D accuracy across the whole path. For each experiment, the absolute difference between the goal position and the position of the POM was evaluated at every control step. Errors were calculated for the x, y, and z positions. The mean and standard deviation of the errors in each axis were computed between the actual and goal paths in each experiment. The errors of the writing experiments are included in Table II. These sub-millimeter error metrics show that the estimated inverse Jacobians are close enough to the true inverse Jacobians to control the system accurately.



(a) (b) (c) (d)

Fig. 5: Shape writing experiments, POM path in green, goal path in red; 5a: Small heart (5.4 x 5.2mm) drawn with Blue Prism; 5b: Large heart (12.1 x 12.4mm) with Blue Prism; 5c: Star drawn by off-center POM with yellow pear; 5d: Brush manipulated in brushing pattern.

TABLE II: Experimental position error

Experiment	X Position Error	Y Position Error	Z Position Error
I	0.773 ± 0.406	0.272 ± 0.192	0.255 ± 0.115
C	0.357 ± 0.243	0.098 ± 0.084	0.271 ± 0.176
R	0.358 ± 0.238	0.397 ± 0.253	0.325 ± 0.239
A	0.511 ± 0.27	0.207 ± 0.13	0.242 ± 0.108
Big Heart	0.515 ± 0.343	0.341 ± 0.189	0.215 ± 0.121
Small Heart	0.314 ± 0.19	0.212 ± 0.192	0.174 ± 0.099
Star	0.787 ± 0.454	0.3 ± 0.19	0.199 ± 0.116
Brush	0.456 ± 0.285	0.771 ± 0.258	0.23 ± 0.175

D. Writing Test Waypoint Error

Additionally, the waypoint errors in Fig. 4 were evaluated to compare against the previous particle filter-based approach. This evaluates errors at the end of a path segment instead of at each control step, as was done in Section IV-C. Errors were calculated for the x, y, and z translation errors, and the L2 norm of these errors was computed at each waypoint. The error of all waypoints in the "ICRA" experiment was 0.453 ± 0.242mm. The handwriting experiment in [18] writing "SCIENCE" with the same path sizes was 0.42 ± 0.34mm. The minor difference in error shows that removing the model doesn't significantly reduce accuracy.

E. Processing Frequency

A significant advantage of this method is the use of efficient linear algebra techniques, enabling fast runtimes. Two filter variations were evaluated for their frequency, one implemented only on the CPU and another with all possible computations run on the GPU. Experiments were performed on a Desktop with an Intel 11700 and NVIDIA 3060. Motor movements and POM pose changes were precalculated in simulation to characterize the filter frequency. Runtime was evaluated from before initial particles were generated until the filter completed 10,000 iterations. The number of filter iterations was divided by the total runtime to calculate the frequency. This process was repeated 20 times, and the filter frequency was averaged to ensure representative statistics.

As shown in Table III, the filter reaches 74.92 ± 2.76Hz on the CPU. On the GPU, the filter reaches 932.64 ± 8.43Hz. The particle filter presented in [18] requires 6 seconds per filter iteration (∼0.1666 Hz) when run on an AMD Ryzen Threadripper 1950X with 32 threads. The most direct comparison is the CPU version of the filter, which takes 0.0133 seconds per filter iteration. The model-free method presented here runs considerably faster than the VLR particle filter. The filter runs quickly enough that systems can update their controller in real time without significantly impacting their control frequency. This is especially important in manipulation, where the inverse Jacobian may vary during tasks, requiring updated estimation. These situations require updating the inverse Jacobian in real time. So, the particle filter's fast run time is essential in these cases.

TABLE III: Particle filter frequencies

Filter type	Frequency (Hz)
Inverse Jacobian CPU	74.92 ± 2.76
Inverse Jacobian GPU	932.64 ± 8.43
VLR Particle Filter CPU [17], [18]	∼0.1666

V. Conclusion

We present a method to perform model-free in-hand manipulation using inverse Jacobians estimated with particle filters. By precomputing a distribution of inverse Jacobians, this method can efficiently represent the space of hand-object systems, which can be sampled to estimate the inverse Jacobian online. This is advantageous as it only requires a simulation of the hand system and no *a priori* knowledge of the object being manipulated. The estimation requires only a single in-palm camera, reducing sensing requirements compared to traditional in-hand manipulation systems. This method can also run at high frequencies, which allows it to quickly update itself if the underlying system changes during manipulation. The particle filter was evaluated in various handwriting tests with different object shapes, sizes, and POM locations, validating the system's adaptability.

In the future, we will evaluate different filter types to remove the requirement for a precomputed distribution of possible inverse Jacobians. Furthermore, we will evaluate the method on three-finger hands with varying finger lengths and contact locations. Additionally, we will apply the system to hands with different numbers of fingers to extend the method past the Model O hand that experiments were performed on. We also want to demonstrate this system on cameras outside the palm by estimating the transform applied to the inverse Jacobian. This would allow the system to perform a broader range of tasks that require tracking with external cameras.

VI. Acknowledgements

We thank Vatsal V. Patel for his insightful discussions.

References

[1] A. Bicchi, "Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 652–662, 2000.

[2] D. J. Montana, "Contact stability for two-fingered grasps," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 4, pp. 421–430, 1992.

[3] J. C. Trinkle, J.-S. Pang, S. Sudarsky, and G. Lo, "On dynamic multi-rigid-body contact problems with coulomb friction," *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 77, no. 4, pp. 267–279, 1997.

[4] T. Okada, "Computer control of multijointed finger system for precise object-handling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 12, no. 3, pp. 289–299, 1982.

[5] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[6] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[7] J. Bütepage, S. Cruciani, M. Kokic, M. Welle, and D. Kragic, "From visual understanding to complex object manipulation," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 161–179, 2019.

[8] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[9] D. Navarro-Alarcon, Y.-H. Liu, J. G. Romero, and P. Li, "Model-free visually servoed deformation control of elastic objects by robot manipulators," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1457–1468, 2013.

[10] D. Berenson, "Manipulation of deformable objects without modeling and simulating deformation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 4525–4532.

[11] B. Calli and A. M. Dollar, "Vision-based precision manipulation with underactuated hands: Simple and effective solutions for dexterity," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1012–1018.

[12] A. Sintov, A. S. Morgan, A. Kimmel, A. M. Dollar, K. E. Bekris, and A. Boularias, "Learning a state transition model of an underactuated adaptive hand," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1287–1294, 2019.

[13] P. Chanrungmaneekul, K. Ren, J. Grace, K. Hang, and A. M. Dollar, "Non-parametric self-identification and model predictive control of dexterous in-hand manipulation," 2023, accepted.

[14] A. S. Morgan, K. Hang, and A. M. Dollar, "Object-agnostic dexterous manipulation of partially constrained trajectories," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5494–5501, 2020.

[15] A. S. Morgan, B. Wen, J. Liang, A. Boularias, A. M. Dollar, and K. Bekris, "Vision-driven compliant manipulation for reliable, high-precision assembly tasks," *arXiv preprint arXiv:2106.14070*, 2021.

[16] A. Sieler and O. Brock, "Dexterous soft hands linearize feedback-control for in-hand manipulation," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 8757–8764.

[17] K. Hang, W. G. Bircher, A. S. Morgan, and A. M. Dollar, "Hand–object configuration estimation using particle filters for dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 14, pp. 1760–1774, 2020.

[18] ——, "Manipulation for self-identification, and self-identification for better manipulation," *Science Robotics*, vol. 6, no. 54, p. eabe1321, 2021.

[19] A. M. Dollar and R. D. Howe, "The highly adaptive sdm hand: Design and performance evaluation," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 585–597, 2010.

[20] L. U. Odhner, R. R. Ma, and A. M. Dollar, "Open-loop precision grasping with underactuated hands inspired by a human manipulation strategy," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 625–633, 2013.

[21] L. U. Odhner, L. P. Jentoft, M. R. Claffee, N. Corson, Y. Tenzer, R. R. Ma, M. Buehler, R. Kohout, R. D. Howe, and A. M. Dollar, "A compliant, underactuated hand for robust manipulation," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 736–752, 2014.

[22] V. V. Patel, D. Rakita, and A. M. Dollar, "An analysis of unified manipulation with robot arms and dexterous hands via optimization-based motion synthesis," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8090–8096.

[23] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[24] A. S. Morgan, K. Hang, W. G. Bircher, and A. M. Dollar, "A data-driven framework for learning dexterous manipulation of unknown objects," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 8273–8280.

[25] C. Corcoran and R. Platt, "A measurement model for tracking hand-object state during dexterous manipulation," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 4302–4308.

[26] R. Labbe, "Kalman and bayesian filters in python," *Chap*, vol. 7, no. 246, p. 4, 2014.

[27] A. Banerjee and P. Burlina, "Efficient particle filtering via sparse kernel density estimation," *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2480–2490, 2010.

[28] K. Hosoda and M. Asada, "Versatile visual servoing without knowledge of true jacobian," in *1994 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 1994, pp. 186–193 vol.1.

[29] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3400–3407.

[30] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.