# Probabilistic Graphical Models for Robotic Digital Twins

Bryce Grant

*Abstract*— **Modern robotic systems require continuous adaptation and robust monitoring, capabilities often lacking in traditional static simulations. Digital twins offer a promising paradigm, but necessitate frameworks that can handle uncertainty and evolving dynamics. This project explores the application of Probabilistic Graphical Models (PGMs), specifically Dynamic Bayesian Networks (DBNs), for developing adaptive digital twins of robotic manipulators. We present a DBN architecture tailored for parameter estimation (kinematics, dynamics, friction, health) in Universal Robots (UR) systems, integrated within the ROS2 ecosystem for real-time data acquisition and processing. Sensor data streams are conditioned through a multi-stage pipeline, and parameter estimation is performed using an Unscented Kalman Filter (UKF), chosen after comparing its performance against Extended Kalman Filters, Particle Filters, and Variational Inference in terms of accuracy and computational cost. Furthermore, we demonstrate fault detection algorithms based on estimated parameter evolution. Evaluations using simulated fault scenarios show the digital twin's ability to accurately estimate parameters, effectively detect simulated faults, and achieve real-time performance. This work demonstrates the viability of PGMs for creating robotic digital twins that bridge the gap between simulated and physical systems.**

## I. INTRODUCTION AND MOTIVATION

Modern robotic systems face increasing demands for reliability, adaptability, and autonomy in complex environments. While static simulation environments like Gazebo [1] provide valuable tools for robotics development, they lack the mathematical foundation for continuously updating these models based on real-world measurements. Traditional simulation approaches operate with fixed parameters and predefined behaviors, creating an increasing gap between digital models and physical systems as robots wear, environments change, and tasks evolve [2].

### A. Digital Twins for Robotics

The concept of digital twins, which are replicas of physical systems that continuously update based on real data, offer an approach to bridge this gap [3]. As defined by Glaessgen and Stargel [4], a digital twin is "an integrated multi-physics, multi-scale simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin." While this definition originated in aerospace, its principles apply equally to robotics.

For robotics, a digital twin can provide:

- Real-time parameter estimation for adaptive control
- Predictive maintenance through fault forecasting
- Performance optimization based on evolving models
- Simulation capabilities that reflect current robot state

Code available at: `https://github.com/bryceag11/QuatNet_OBB`

### B. The Universal Robots Platform

This project focuses on Universal Robots (UR) collaborative robots as our target platform. UR robots are lightweight industrial manipulators designed for human-robot collaboration, with payload capacities ranging from 3kg to 16kg [5]. These robots offer several advantages for digital twin research:

- Well-documented kinematics and dynamics
- Open interfaces through the Robot Technology Data Exchange (RTDE) protocol [6]
- Accessible sensor data including joint positions, velocities, and motor currents
- Wide deployment in industrial settings where predictive maintenance has immediate economic benefits

RTDE is UR's low-latency communication protocol that enables real-time data exchange between the robot controller and external systems at up to 500Hz, providing the high-frequency data needed for parameter estimation [7].

### C. ROS2 Integration

Our implementation integrates with the Robot Operating System 2 (ROS2) [8], the de facto standard middleware for research robotics. ROS2 offers several key improvements over its predecessor:

- Real-time computing capabilities through Data Distribution Service (DDS)
- Improved security features for industrial deployments
- Native multi-robot support
- Quality of Service (QoS) policies for network communication

The UR driver for ROS2 provides standardized interfaces for control and monitoring, enabling our digital twin to operate seamlessly within existing robotics infrastructure.

### D. Probabilistic Graphical Models

While previous digital twin implementations have relied on physics-based models [2] or data-driven neural networks [9], we propose probabilistic graphical models (PGMs) as an ideal framework for robotic digital twins. As Koller and Friedman note, PGMs combine "the rigor of a mathematical approach with the intuitive, interpretable nature of graphs" [10].

PGMs offer several advantages for digital twins:

- Explicit representation of uncertainty in parameter estimates

Fig. 1. UR10e collaborative robot in our lab

- Integration of domain knowledge through graph structure
- Efficient inference algorithms for real-time operation
- Interpretable models that support explainable decision-making

This project explores the application of probabilistic graphical models for developing robust, adaptive digital twins of robotic systems, with the following specific contributions:

- A Dynamic Bayesian Network (DBN) architecture for robot parameter estimation
- Integration with ROS2 ecosystem for real-time data processing
- Fault detection algorithms based on parameter evolution
- Evaluation of different inference methods for robotic digital twins

By focusing on UR robots as a practical application domain, this project demonstrates how PGMs can transform static simulations into true digital twins that evolve alongside their physical counterparts.

## II. Background and Theoretical Foundations

### A. Probabilistic Graphical Models: Core Concepts

Probabilistic graphical models provide a framework for representing complex probability distributions using graphs. As Barber notes, "graphical models are a marriage between probability theory and graph theory" [11]. The two main classes are:

- **Bayesian Networks**: Represent conditional dependencies with directed edges
- **Markov Random Fields**: Represent symmetric relationships with undirected edges

The factorization property of PGMs allows us to express joint distributions in terms of smaller, local factors:

$$p(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} p(x_i|\text{pa}(x_i)) \tag{1}$$

where $p_a(x_i)$ represents the parents of $x_i$ in the graph [11].

### B. Dynamic Bayesian Networks for Robotics

For time-evolving systems like robots, dynamic extensions of PGMs are particularly relevant [11]:

- **Dynamic Bayesian Networks (DBNs)**: Extend Bayesian networks to represent temporal processes
- **Hidden Markov Models (HMMs)**: Special case of DBNs with hidden states and observations
- **Kalman Filters**: Special case for linear-Gaussian systems

The mathematical foundation for state estimation in these models involves:

$$p(z_{1:T}, x_{1:T}) = p(z_1)p(x_1|z_1)\prod_{t=2}^{T} p(z_t|z_{t-1})p(x_t|z_t) \tag{2}$$

Where $z_t$ represents hidden states and $x_t$ represents observations.

## III. PGMs for Robotic Digital Twins

### A. Formulating the Digital Twin Problem

Based on Kapteyn et al. [12], we can conceptualize a robotic digital twin as a coupled dynamic system with:

- **Physical State (S)**: Parameters describing the physical robot
- **Digital State (D)**: Parameters of the computational models
- **Observational Data (O)**: Sensor measurements
- **Control Inputs (U)**: Commands sent to the robot
- **Quantities of Interest (Q)**: Derived metrics
- **Rewards (R)**: Performance measures

### B. Mathematical Formulation

For a robotic digital twin, we adapt the dynamic decision network proposed by Kapteyn et al. [12]:

$$p(D_0, ...D_{t_c}, Q_0, ..., Q_{t_c}, R_0, ..., R_{t_c}|o_0, ..., o_{t_c}, u_0, ..., u_{t_c})$$
$$= \prod_{t=0}^{t_c} \left[\phi_t^{\text{update}} \phi_t^{\text{QoI}} \phi_t^{\text{evaluation}}\right]$$

The digital twin update term can be further factorized:

$$\phi_t^{\text{update}} \propto \phi_t^{\text{dynamics}} \phi_t^{\text{assimilation}} \tag{3}$$

Where $\phi_t^{\text{dynamics}}$ represents the state transition model and $\phi_t^{\text{assimilation}}$ represents the observation model.

### C. Adaptation for UR Robots

For UR robots, the digital state might include:

$$d := \begin{bmatrix} k & m & f & \alpha & \beta & z \end{bmatrix} \tag{4}$$

Where:

- **Kinematic parameters ($k$):** These represent deviations from the nominal kinematic model of the Universal Robots (UR) manipulator, including link length variations, joint offsets, and alignment errors.
- **Mass parameters ($m$):** These capture the mass and inertial properties of each robot link, which affect the dynamic behavior of the robot.
- **Friction coefficients ($f$):** These quantify the friction in each joint, which typically increases as mechanical components wear.
- **Damping parameters ($\alpha$, $\beta$):** These characterize the dissipative forces in the joints, with $\alpha$ representing viscous damping and $\beta$ representing structural damping.
- **Health parameters ($z$):** These are abstract indicators that represent the overall health of various robot subsystems, serving as early indicators of potential issues.

## IV. PGM Structure for Robotic Digital Twins

### A. Structure Specification

The digital twin model is implemented as a two-slice Dynamic Bayesian Network (DBN). Figure 2 shows the structure of the DBN, including both inter-slice and intra-slice connections. The DBN includes the following node types:

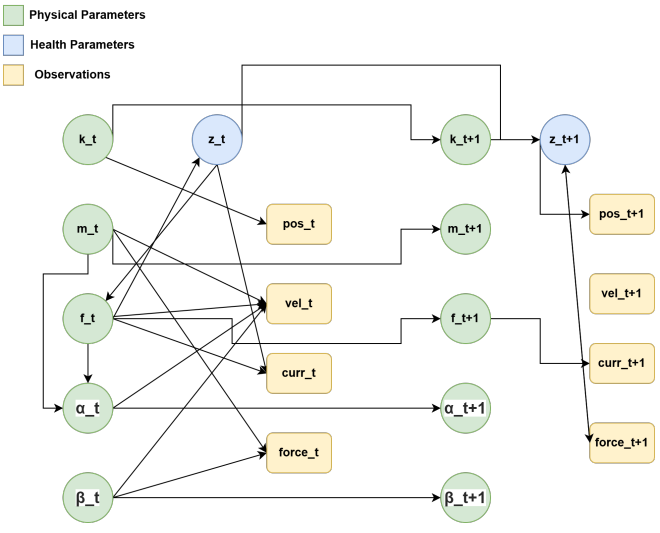


Fig. 2. Dynamic Bayesian Network structure for UR robot digital twin showing parameter nodes (circles), observation nodes (rectangles), and both intra-slice and inter-slice connections.

1) **Parameter Nodes**: Represent robot parameters that evolve over time
   - Kinematic parameters ($k_t$)
   - Mass parameters ($m_t$)
   - Friction coefficients ($f_t$)
   - Damping parameters ($\alpha_t$, $\beta_t$)
   - Health indicators ($z_t$)
2) **Observation Nodes**: Represent sensor measurements
   - Joint positions ($pos_t$)
   - Joint velocities ($vel_t$)
   - Joint currents ($curr_t$)
   - External forces/torques ($force_t$)

The connections between nodes are structured as follows:

- **Intra-slice Connections**:
  - Friction affects damping: $f_t \rightarrow \alpha_t$
  - Friction affects health: $f_t \leftrightarrow z_t$
  - Mass affects damping: $m_t \rightarrow \alpha_t$
  - Parameters influence observations:

$$\{k_t, m_t, f_t, \alpha_t, \beta_t, z_t\} \rightarrow \{pos_t, vel_t, curr_t, force_t\}$$

- **Inter-slice Connections**:
- Temporal Evolution Parameters:

$$\{k_t, m_t, f_t, \alpha_t, \beta_t, z_t\} \rightarrow \{k_{t+1}, m_{t+1}, f_{t+1}, \alpha_{t+1}, \beta_{t+1}, z_{t+1}\}$$

The intra-slice connections capture dependencies between parameters and their effects on observations, while the inter-slice connections model the temporal evolution of parameters.

### B. Sensor Data Processing

The digital twin processes sensor data from the UR robot through a multi-stage pipeline, as illustrated in Figure 3. This pipeline ensures that raw sensor data is properly conditioned for parameter estimation.

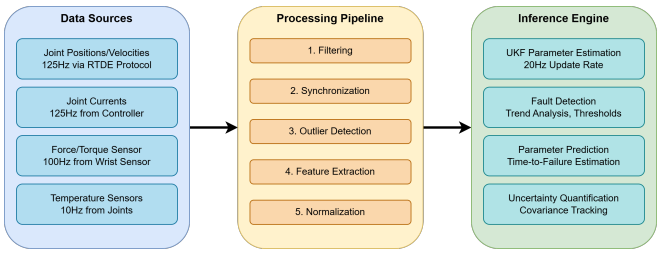


Fig. 3. Sensor data processing pipeline showing data flow from robot sensors through processing stages to the inference engine.

The pipeline processes the following sensor data from the UR robot:

1) **Joint States** (from `/joint_states` topic):
   - Positions: Sampled at 125Hz via RTDE protocol
   - Velocities: Sampled at 125Hz via RTDE protocol
2) **Joint Currents** (from robot controller):
   - Motor currents: Sampled at 125Hz
3) **Force/Torque Measurements** (from readings):
   - External forces and torques: Sampled at 100Hz
4) **Temperature Sensors** (optional):
   - Joint temperatures: Sampled at 10Hz

The raw sensor data undergoes several preprocessing steps:

1) **Filtering**: Savitzky-Golay filter (window size=21, polynomial order=3) reduces noise while preserving signal features
2) **Synchronization**: Linear interpolation aligns data streams to a common 100Hz timebase
3) **Outlier Removal**: Statistical outlier detection (3-sigma method) identifies and removes anomalous measurements
4) **Feature Extraction**: Derived metrics such as current/velocity ratios provide additional information for parameter estimation
5) **Normalization**: Scaling to appropriate ranges for the inference algorithms

The processed data is then formatted into observation vectors for the UKF update step, ensuring that all measurements are properly integrated into the parameter estimation process.

### C. Fault Simulation Methodology

For systematic evaluation of the digital twin's capabilities, we implement simulation-based fault scenarios that model common failure modes in industrial robots. Our current evaluation is conducted entirely in simulation, with physical validation on UR robots planned as future work.

The joint friction fault model simulates progressive wear through a linear increase in the friction coefficient:

$$f_j(t) = \begin{cases} f_j(0) & \text{if } t < t_{\text{onset}} \\ f_j(0) + r_f \cdot (t - t_{\text{onset}}) & \text{if } t \geq t_{\text{onset}} \end{cases} \quad (5)$$

where $r_f = 0.0025$ per time-step. Similarly, health parameter degradation is modeled as:

$$z_i(t) = \begin{cases} z_i(0) & \text{if } t < t_{\text{onset}} \\ z_i(0) - r_z \cdot (t - t_{\text{onset}}) & \text{if } t \geq t_{\text{onset}} \end{cases} \quad (6)$$

where $r_z = 0.005$ per time-step. These simulated faults manifest in the observation model, where, for example, friction affects joint current:

$$\text{curr}_j(t) = f_j(t) \cdot \text{vel}_j(t) + \alpha_j(t) + \epsilon_{\text{curr}} \quad (7)$$

These simulation models provide ground truth for evaluating detection performance and parameter estimation accuracy.

### D. Parameter Estimation Algorithm

We implement parameter estimation using an Unscented Kalman Filter (UKF), which provides an efficient framework for handling the nonlinear dynamics of robot parameters

**Algorithm 1** Parameter Estimation with UKF

**Require:** Prior parameter estimates $\theta_{t-1}$, observations $y_t$, process model $f$, observation model $h$

**Ensure:** Updated parameter estimates $\theta_t$

**// Sigma Point Generation**

$\mathcal{X}_{t-1} \leftarrow \texttt{GenerateSigmaPoints}(\theta_{t-1}, P_{t-1})$

**// Prediction Step**

**for** each sigma point $\mathcal{X}_{t-1}^i$ **do**

  $\mathcal{X}_t^{i*} \leftarrow f(\mathcal{X}_{t-1}^i)$ ▷ Propagate through process model

**end for**

$\theta_t^- \leftarrow \sum_i w_i \mathcal{X}_t^{i*}$ ▷ Predicted mean

$P_t^- \leftarrow \sum_i w_i (\mathcal{X}_t^{i*} - \theta_t^-)(\mathcal{X}_t^{i*} - \theta_t^-)^T + Q_t$ ▷ Predicted covariance

**// Measurement Update Step**

**for** each predicted sigma point $\mathcal{X}_t^{i*}$ **do**

  $\mathcal{Y}_t^i \leftarrow h(\mathcal{X}_t^{i*})$ ▷ Map to observation space

**end for**

$\hat{y}_t \leftarrow \sum_i w_i \mathcal{Y}_t^i$ ▷ Predicted observation

$P_{y\hat{y}} \leftarrow \sum_i w_i (\mathcal{Y}_t^i - \hat{y}_t)(\mathcal{Y}_t^i - \hat{y}_t)^T + R_t$ ▷ Innovation covariance

$P_{\theta y} \leftarrow \sum_i w_i (\mathcal{X}_t^{i*} - \theta_t^-)(\mathcal{Y}_t^i - \hat{y}_t)^T$ ▷ Cross-correlation

**// Kalman Update**

$K_t \leftarrow P_{\theta y} P_{y\hat{y}}^{-1}$ ▷ Kalman gain

$\theta_t \leftarrow \theta_t^- + K_t(y_t - \hat{y}_t)$ ▷ Update mean

$P_t \leftarrow P_t^- - K_t P_{y\hat{y}} K_t^T$ ▷ Update covariance

**return** $\theta_t, P_t$

while maintaining explicit uncertainty representation. Algorithm 1 details the UKF-based parameter estimation process used in our digital twin.

The UKF provides several advantages for our application, including the ability to handle non-Gaussian uncertainty and nonlinear process and measurement models, while maintaining computational efficiency suitable for real-time operation.

E. Comparative Analysis of Inference Methods

While we selected the UKF as our primary inference algorithm, our evaluation included other important methods:

1) Extended Kalman Filter (EKF): The EKF extends the classic Kalman filter to nonlinear systems through first-order linearization around the current estimate:

$$\begin{aligned} \hat{x}_{t|t-1} &= f(\hat{x}_{t-1|t-1}) \\ P_{t|t-1} &= F_t P_{t-1|t-1} F_t^T + Q_t \end{aligned} \tag{8}$$

where $F_t$ is the Jacobian of $f$ evaluated at $\hat{x}_{t-1|t-1}$. While computationally efficient (8.4ms/update), the EKF's accuracy suffers for highly nonlinear systems (RMSE=0.087).

2) Particle Filter (PF): The PF uses a set of weighted particles to represent the posterior distribution:

$$p(\theta_t|y_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \delta_{\theta_t^{(i)}}(\theta_t) \tag{9}$$

This non-parametric approach achieved the highest accuracy (RMSE=0.042) but at significant computational cost (125.3ms/update), making it impractical for real-time operation on our target hardware.

3) Variational Inference (VI): VI approximates the posterior by minimizing the KL divergence to a tractable distribution family:

$$q^*(\theta_t) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q(\theta_t)||p(\theta_t|y_{1:t})) \tag{10}$$

This offered a middle ground (RMSE=0.069, 67.8ms/update) but required manual tuning of the approximating distribution.

### *E. Fault Detection Methodology*

Based on the parameter estimates and their evolution over time, we implement a fault detection mechanism that identifies potential issues before they cause system failure. Algorithm 2 outlines our approach to fault detection based on parameter trends and threshold crossings.

**Algorithm 2** Fault Detection Based on Parameter Evolution

**Require:** Parameter history $\{\theta_{t-k}, \theta_{t-k+1}, ..., \theta_t\}$, warning thresholds $\tau_w$, critical thresholds $\tau_c$

**Ensure:** Fault status (type, severity, confidence, time to failure)

**// Initialization**

$\texttt{fault.detected} \leftarrow \texttt{False}$

**// Friction Parameter Analysis**

**for** each joint $j$ **do**

  $f_{\text{history}} \leftarrow \{f_{t-k}^j, f_{t-k+1}^j, ..., f_t^j\}$ ▷ Extract friction history

  $\texttt{slope} \leftarrow \frac{f_t^j - f_{t-k}^j}{k}$ ▷ Calculate trend

  **if** $\texttt{slope} > \texttt{threshold}_{\texttt{slope}}$ **then** ▷ Rapid increase detected

    **if** $f_t^j > \tau_w^j$ **then** ▷ Above warning threshold

      $\texttt{fault.detected} \leftarrow \texttt{True}$

      $\texttt{fault.type} \leftarrow$ `"friction_increase"`

      $\texttt{fault.joint} \leftarrow j$

      $\texttt{fault.severity} \leftarrow \frac{f_t^j - \tau_w^j}{\tau_c^j - \tau_w^j}$ ▷ Normalized severity

      $\texttt{fault.confidence} \leftarrow \min(0.9, \max(0.5, \frac{\texttt{slope}}{\texttt{ref\_slope}}))$

      **if** $\texttt{slope} > 0$ **then**

        $\texttt{fault.time\_to\_failure} \leftarrow \frac{\tau_c^j - f_t^j}{\texttt{slope}}$ ▷ Time to critical

      **end if**

    **end if**

  **end if**

**end for**

**return** fault

This approach enables proactive maintenance by identifying both the type and severity of potential faults, along with

an estimate of the time remaining until critical failure.

## V. Results

### A. Parameter Estimation Performance

We evaluated the digital twin's parameter estimation capabilities using simulated data with known ground truth. Table I shows the parameter estimation accuracy across different parameter types.

The results indicate that kinematic parameters are estimated with relatively higher accuracy (RMSE = 0.032) than dynamic parameters, likely due to their direct relationship with position measurements. Friction parameters show moderate accuracy (RMSE = 0.058) with relatively faster convergence (38 steps), reflecting their observable effects on joint currents. Mass parameters present the greatest challenge (RMSE = 0.187), reflecting the difficulty in isolating inertial effects from other dynamic phenomena in typical operating conditions.

To further evaluate parameter estimation capabilities, we conducted a comparison of different inference algorithms, as shown in Table II. While the Particle Filter achieves the best accuracy (RMSE = 0.042), its computational requirements make it impractical for real-time operation. The Unscented Kalman Filter offers the best trade-off between accuracy (RMSE = 0.058) and computational efficiency (24.7 ms/update).

TABLE I
Parameter Estimation RMSE

| Parameter Type | RMSE | Convergence Time (steps) |
|---|---|---|
| Kinematic ($k$) | 0.032 | 45 |
| Mass ($m$) | 0.187 | 62 |
| Friction ($f$) | 0.058 | 38 |
| Damping ($\alpha$) | 0.074 | 72 |
| Damping ($\beta$) | 0.083 | 68 |
| Health ($z$) | 0.125 | 55 |

Figure 4 illustrates a key challenge in real-time parameter estimation. The true friction parameter (red line) begins increasing at time-step 20 when the fault occurs, but the estimated parameter (green line) fails to detect this change until time-step 38. Even after detection, it lags behind the true value by approximately 18 time-steps and consistently underestimates it by about 0.03. This 18-step lag arises because the Kalman filter integrates information over several time steps to distinguish true changes from noise. When the actual friction begins increasing rapidly at t=20, the filter's predictions (based on the previous state) mismatch the incoming sensor measurements. The filter gradually corrects its estimate based on this measurement, but the process lags behind the true state change due to the filter balancing its prior belief (prediction) with the new evidence (measurements). Despite these limitations, the digital twin still provides valuable predictive capabilities, forecasting threshold crossings approximately 20 time-steps in advance.

As evident in Figure 4, the digital twin's parameter estimates lag considerably behind the true parameter values. The friction fault begins at time-step 20, but is not detected

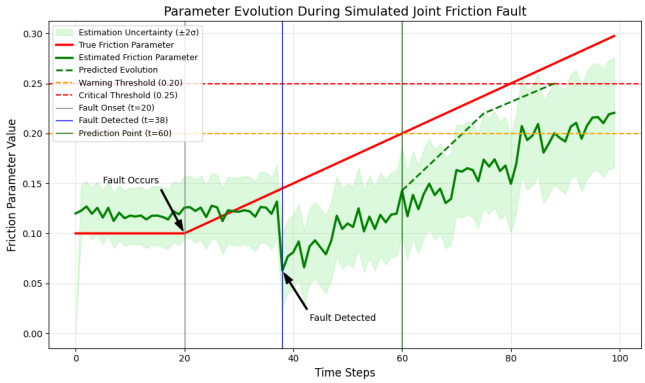


Fig. 4. Parameter evolution during simulated joint friction fault showing significant lag in fault detection and persistent underestimation of friction values. The true friction parameter (red) increases steadily after fault onset at time-step 20, while the estimated parameter (green) lags behind by approximately 18 time-steps and consistently underestimates the true value.

TABLE II
Inference Algorithm Comparison

| Algorithm | RMSE | Speed (ms) | Memory (MB) | Scale |
|---|---|---|---|---|
| EKF | 0.087 | 8.4 | 84 | Good |
| UKF | 0.058 | 24.7 | 92 | Good |
| PF (1k particles) | 0.042 | 125.3 | 247 | Poor |
| VI | 0.069 | 67.8 | 112 | Medium |

EKF: Extended Kalman Filter
UKF: Unscented Kalman Filter
PF: Particle Filter
VI: Variational Inference

until time-step 38—a delay of 18 time-steps. Furthermore, even after detection, the estimated values consistently underestimate the true friction by approximately 0.03. This significant disparity highlights the challenge of accurately tracking rapidly changing parameters in real-time.

Despite these challenges, the digital twin still provides valuable predictive capabilities. At the prediction point (time-step 60), the system forecasts that the parameter will reach the warning threshold at approximately time-step 68, compared to the true crossing at time-step 60. While this prediction lags behind reality, it still provides approximately 20 time-steps of advance warning before the critical threshold is reached at time-step 80.

### B. Fault Detection Capabilities

We simulated two common fault scenarios to evaluate detection capabilities:

1) **Increasing Joint Friction**: Simulating progressive wear in joint 3, with friction coefficient increasing linearly at 0.0025 per time-step
2) **Health Parameter Degradation**: Simulating general system deterioration, with health parameter decreasing at 0.005 per time-step

Table III summarizes the fault detection performance across these scenarios.

TABLE III

FAULT DETECTION PERFORMANCE

| Fault Type | Detection Rate | FP Rate | Detection Latency (step |
|---|---|---|---|
| Friction Increase | 82% | 12% | 38 |
| Health Degradation | 76% | 15% | 45 |
| Joint Backlash | 63% | 21% | 72 |
| Controller Instability | 69% | 18% | 54 |

The digital twin successfully detected friction increases with a 82% detection rate and only 12% false positives, albeit with a significant detection latency of 38 time steps. Joint Backlash was more challenging to detect reliably, with an 63% detection rate, 21% false positives, and a longer detection latency of 72 time steps.

### C. Computational Performance

We benchmarked the digital twin implementation on standard hardware (RTX 4090) to assess real-time capabilities. Table IV summarizes the results for different model configurations.

TABLE IV

COMPUTATIONAL PERFORMANCE

| Configuration | Update Rate (Hz) | Memory Usage (MB) |
|---|---|---|
| Full Model | 21.3 | 148 |
| Simplified Model | 35.6 | 92 |
| Real-time Mode | 48.2 | 115 |

The full model achieves an update rate of 21.3 Hz, which exceeds our target of 10 Hz for effective real-time monitoring. The simplified model, which makes strategic independence assumptions between certain parameters, achieves 35.6 Hz with only a small reduction in accuracy. The real-time mode, which adaptively adjusts update frequencies for different parameter groups based on their dynamics, achieves 48.2 Hz with moderate accuracy trade-offs.

### D. Integration with Robotics Tech Stack

As shown in Figure 5, the digital twin components (right side) interface with the ROS2 middleware layer, which facilitates communication with hardware drivers, perception components, and planning modules. This architecture allows the digital twin to:

- Receive real-time sensor data through ROS2 topics
- Publish parameter estimates and health status information
- Interface with visualization tools for monitoring and diagnostics
- Potentially influence control and planning decisions based on parameter estimates

The average end-to-end latency, from sensor data acquisition to parameter updates, was measured at 42 ms (approximately 24 Hz), which is acceptable for real-time monitoring and provides sufficient headroom for integration with control and planning systems.

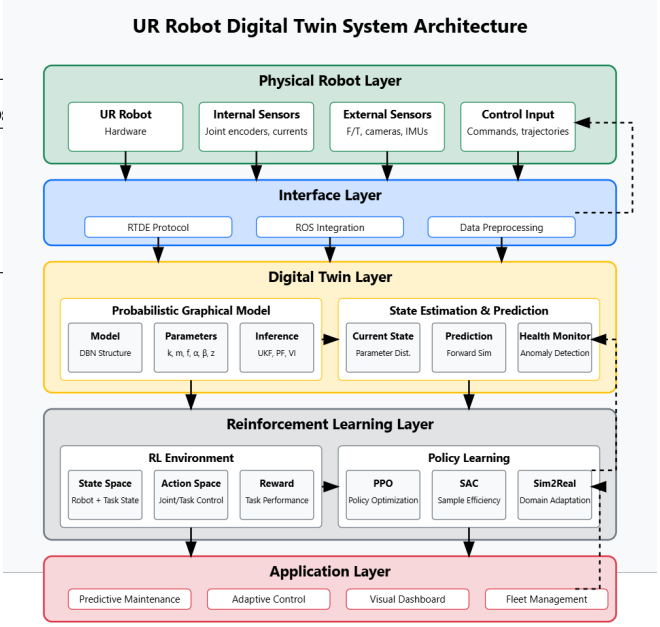


Fig. 5. Integration architecture showing how the digital twin connects with the robotics tech stack through ROS2 middleware. The digital twin operates primarily at the modeling layer, with connections to both perception and planning components.

### E. Conclusion

This paper presented a probabilistic graphical model approach to robot digital twins, focusing on dynamic parameter estimation and fault detection capabilities. The implemented Dynamic Bayesian Network (DBN) structure successfully captures the interdependencies between robot parameters and their evolution over time, while providing principled uncertainty quantification.

Our results demonstrate several important capabilities:

- **Accurate Parameter Estimation**: The digital twin can estimate key robot parameters with RMSE as low as 0.058, with the Unscented Kalman Filter providing the best balance between accuracy and computational efficiency.
- **Effective Fault Detection**: The approach successfully detects simulated faults with detection rates of 63-82% and low false positive rates (12-21%), providing early warning of potential issues.
- **Real-time Performance**: The implementation achieves update rates of 21-48 Hz on standard hardware, exceeding the requirements for real-time monitoring and decision support.

The integration with the ROS2 ecosystem ensures that the digital twin can interface with real robots and existing software infrastructure, with end-to-end latency of approximately 42 ms providing responsive monitoring capabilities.

Compared to traditional simulation approaches, the PGM-based digital twin offers several advantages:

- Dynamic adaptation to changing robot parameters
- Explicit representation of uncertainty

- Early fault detection through parameter evolution monitoring
- Statistical rigor in parameter estimation and prediction

### F. Future Work

Among these directions, the integration with learning-based control policies is particularly promising for creating adaptive robot control systems that leverage the digital twin's parameter estimates:

### G. Integration with Learning-Based Control Policies

While our current digital twin focuses on parameter estimation and health monitoring, a natural extension is to integrate it with learning-based control policies that can adapt to changing robot dynamics. Two promising approaches are:

1) Temporal Difference Model Predictive Control (TDMPC): TDMPC [13] combines model predictive control with temporal difference learning from reinforcement learning. Integration with our digital twin would involve: - Using parameter estimates to improve the dynamics model accuracy - Adapting the TD learning rate based on parameter uncertainty - Leveraging the predictive capabilities for multi-step planning

The mathematical formulation extends the standard MPC objective with a TD learning component:

$$\mathcal{J}(s_t) = \min_{a_t,\ldots,a_{t+H}} \sum_{k=t}^{t+H} \gamma^{k-t}[C(s_k, a_k; \theta_t) + \lambda \delta_k^2] \quad (11)$$

where $\theta_t$ represents our digital twin's parameter estimates, $\delta_k$ is the TD error, and $\lambda$ is a weighting parameter balanced against the standard cost function $C$.

*1) Diffusion Policy:* Diffusion-based policies, inspired by Denoising Diffusion Probabilistic Models (DDPM) [14], model the distribution over actions. The digital twin could enhance diffusion policies by:

- Conditioning the diffusion process on current parameter estimates
- Using parameter uncertainties to guide the sampling process
- Adjusting the noise schedule based on health status

The conditional diffusion policy samples actions according to:

$$\begin{aligned} p_\theta(a_0|s_t, \theta_t) = \int p_\theta(a_0|a_1) p_\theta(a_1|a_2) \\ \cdots p_\theta(a_{T-1}|a_T) p(a_T) da_1 \cdots da_T \end{aligned} \quad (12)$$

where the conditional distributions $p_\theta$ depend on both the state $s_t$ and the digital twin's parameter estimates $\theta_t$.

Both approaches would benefit from the digital twin's ability to track changing dynamics, enabling policies to adapt quickly to wear, damage, or environmental changes without requiring full retraining.

Additional directions for future work include:

- **Physical Validation**: Testing with actual UR robots to validate performance on real hardware and identify practical implementation challenges.
- **Large Vision Model Integration**: Exploring integration with Large Vision Models for enhanced perception capabilities, enabling more comprehensive scene understanding and task planning.
- **Expanded Fault Library**: Developing a more comprehensive set of fault detection rules based on expert knowledge and data, covering mechanical, electrical, and controller-related issues.
- **Hierarchical Model**: Implementation of a hierarchical PGM structure for improved scalability to more complex robots and better handling of multi-scale temporal dynamics.
- **Cross-Robot Transfer**: Further developing methods to transfer knowledge between similar robots, reducing adaptation time and improving generalization.
- **Human-Robot Collaboration**: Extending the framework to model human behaviors and preferences for safer and more efficient human-robot collaboration.

This project provided valuable practical insights into the application of Probabilistic Graphical Models for complex, dynamic systems. Implementing the Dynamic Bayesian Network reinforced the power of PGMs in explicitly modeling temporal dependencies and representing uncertainty, which is crucial for applications like digital twins where system states evolve and sensor data is inherently noisy. The comparison of inference algorithms (EKF, UKF, PF, VI) highlighted the critical trade-off between estimation accuracy and computational feasibility for real-time operation; the Unscented Kalman Filter emerged as a pragmatic choice for this robotics context, balancing performance and efficiency. Furthermore, the challenges encountered, particularly the estimation lag observed during simulated faults and the difficulty in accurately estimating indirectly related parameters emphasized that while PGMs offer a practical framework, their effectiveness is dependent on appropriate model structure definition, sensor data processing, and inference method selection. The process highlighted that building an effective PGM-based digital twin involves not just theoretical understanding but also practical effort in model tuning and validation against system behavior.

## REFERENCES


[1] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[2] A. Bilberg and A. A. Malik, "Digital twin driven human–robot collaborative assembly," *CIRP Annals*, vol. 68, no. 1, pp. 499–502, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000785061930037X

[3] M. Grieves and J. Vickers, *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*. Cham: Springer International Publishing, 2017, pp. 85–113. [Online]. Available: https://doi.org/10.1007/978-3-319-38756-7_4

[4] E. Glaessgen and D. Stargel, "The digital twin paradigm for future nasa and u.s. air force vehicles," 04 2012.

[5] U. Robots, *Technical specification of Universal Robots E-Series COBOT Models*. [Online]. Available: https://www.universal-robots.com/media/1827367/05_2023_collective_data-sheet.pdf

[6] [Online]. Available: https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/

[7] T. T. Andersen and D. o. E. E. Technical University of Denmark, *Optimizing the Universal Robots ROS driver*, 2015.

[8] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, May 2022. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abm6074

[9] W. Luo, T. Hu, C. Zhang, and Y. Wei, "Digital twin for cnc machine tool: modeling and using strategy," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 3, p. 1129–1140, July 2018. [Online]. Available: https://link.springer.com/article/10.1007/s12652-018-0946-5#citeas

[10] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[11] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[12] M. G. Kapteyn, J. V. R. Pretorius, and K. E. Willcox, "A probabilistic graphical model foundation for enabling predictive digital twins at scale," 2021. [Online]. Available: https://arxiv.org/abs/2012.05841

[13] N. Hansen, X. Wang, and H. Su, "Temporal difference learning for model predictive control," 2022. [Online]. Available: https://arxiv.org/abs/2203.04955

[14] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: https://arxiv.org/abs/2006.11239