Detecting Anomalous Robot Motion in Collaborative Robotic Manufacturing Systems

Yuhao Zhong, *Student Member, IEEE*, Yalun Wen, Sarah Hopko, Adithyaa Karthikeyan, Prabhakar Pagilla, Ranjana K. Mehta, *Member, IEEE*, and Satish T.S. Bukkapatnam

***Abstract*—Anomalous robot motions caused by cyber attacks and inherent defects can lead to task failures as well as harmful accidents in collaborative human-robot workplaces. External Internet-of-Things (IoT) sensors are essential for reliably monitoring robot tool paths and detecting deviations as early as possible, especially in anomalous situations where the robot system and its internal sensors may not function as expected. However, affordable external IoT sensors may suffer from poor-quality measurements, necessitating data-driven algorithmic enhancements. In addition to external sensors providing independent monitoring, to effectively detect trajectory deviations, we need anomaly detection methods that can capture complex dynamic patterns in the nonlinear trajectory time-series data. In this work, we propose a framework to accurately estimate robot trajectories during normal robot operations as well as to detect various types of anomalous trajectory changes using an external camera. The proposed framework efficiently incorporates marker-based pose estimation, Long Short-Term Memory (LSTM), and residual control charts. The framework was evaluated in a shared human-robot assembly task. The results show that we can accurately estimate robot trajectories by enhancing camera-based measurements. Moreover, it effectively detects anomalous trajectory changes in their early stages. The motion deviation upon detection assists in determining a safe working distance. The framework is also generalizable to previously unseen trajectory deviations and allows the use of a variety of IoT sensors.**

***Index Terms*—External sensors, LSTM (Long Short-Term Memory), Marker-based pose estimation, Residual control chart, Robot trajectory change detection**

I. Introduction

COLLABORATIVE robots have been increasingly employed in the manufacturing industry by integrating human-robot collaboration, which leverages advantages of manual labor (e.g., flexibility, adaptivity, situation awareness) and that of robots (e.g., productivity, capability, and repetitivity) [1], [2]. Moreover, recent global disruptions, such as COVID-19, have seen an annual surge of nearly 40% in the adoption of Industrial Internet of Things (IIoT) and artificial intelligence (AI) within manufacturing systems [3]. This empowers collaborative robots to continuously advance their operational capabilities through data collected by IIoT sensors and processed with AI [4], [5].

However, emerging safety and security concerns may impede the broader adoption of collaborative robots [6]–[9]. In particular, anomalous robot motions can lead to poor performance as well as potential for accidents, including disruptions of critical tasks, property damage, and, even worse, casualties. The anomalous motions can occur not only due to inherent mechanical degradation and faults, but also as a result of cyber attacks. According to [10], sabotage remains the main objective of cyber attacks in manufacturing systems. Recent incidents, such as the Stunext, Norsk Hydro ransomware, and the German steel mill cyberattack, underscore the growing risk of these threats [11], [12]. To prevent such breaches, it is crucial to monitor robot trajectories and detect anomalous motions promptly during the execution of robot tasks.

In general, sensors for monitoring collaborative robots include internal sensors (embedded within the robot hardware) and external sensors (independent of robots) [13]. Under nominal conditions, internal sensors usually provide reliable measurements that are indicative of expected robot motions. However, since the internal sensors share the same control system with the robot, they may malfunction along with the robot. Under certain anomalous circumstances such as cyber attacks, internal sensors can even cause robot malfunctions by passing false information to the controller. For example, attackers can manipulate location feedback to make robots attack ordinary objects or people [14]. Therefore, the use of external IoT sensors (e.g., cameras) becomes vital in such scenarios. These sensors offer remote, independent monitoring of robot motions, providing enhanced reliability and redundant safety in emerging IIoT systems [4], [15], [16].

Nonetheless, few studies have focused on exploiting external sensors to monitor and detect anomalous motions of collaborative robots, considering the following challenges:

1) Existing low-cost external sensors, including IoT devices, are limited by high levels of noise (i.e., inconsistent measurements), inadequate sampling rate (i.e., discontinuous data due to fast robot motions), and low accuracy [17]–[19]. Consequently, these drawbacks compromise the effectiveness of anomaly detection.
2) Even with accurate measurements, conventional anomaly detection methods such as control charts [20] and one-class classification [21] may not be

This work was supported in part by National Science Foundation under Grant 2234972.

Yuhao Zhong, Adithyaa Karthikeyan, Ranjana K. Mehta, and Satish TS Bukkapatnam are with the Department of Industrial & Systems Engineering, Texas A&M University, College Station, TX 77840 USA (e-mail: hirobin_zhong@tamu.edu, adithyaa1996@tamu.edu, rmehta@tamu.edu, satish@tamu.edu).

Yalun Wen, was with the Department of Mechanical Engineering, Texas A&M University, College Station, TX 77840 USA (e-mail: ywen27@tamu.edu).

Sarah Hopko, was with the Department of Industrial & Systems Engineering, Texas A&M University, College Station, TX 77840 USA (e-mail: s.hopko@tamu.edu).

Prabhakar Pagilla, is with the Department of Mechanical Engineering, Texas A&M University, College Station, TX 77840 USA (e-mail: ppagilla@tamu.edu).

effective. Specifically, control charts typically assume the monitored quantity to be stationary or to follow a time-invariant distribution under nominal conditions [20], whereas robot trajectories are usually nonlinear functions of time. On the other hand, the training of one-class classifiers tends to overlook the variation in nominal data, making them less sensitive to subtle changes (i.e., the early stage of an anomaly) and different types of anomalies.

In this work, we propose a framework to address the challenges associated with the use of inexpensive external sensors and anomaly detection based on measurements from those sensors. To improve the large amount of noisy and imprecise sequential robot pose estimates from an external camera, we employ a long short-term memory (LSTM) deep learning model [22]. The LSTM regression model learns to map pose estimates onto the actual robot poses. The model is only trained using data under nominal conditions so that it performs poorly on abnormal data. Then, anomaly detection is achieved by monitoring residuals (or errors) of the model. Residuals tend to be more stationary than nonlinear robot trajectories under nominal conditions, enabling anomaly detection via control charts. Additionally, the regression model enables our framework to effectively handle variations and spatiotemporal patterns present in the trajectory data. Here, internal sensor measurements serve as the ground truth for calculating the residuals. To make the anomaly detection independent of internal sensors for future use, another LSTM is built to directly estimate the first model's residuals using the same input. In summary, our framework consists of trajectory estimation, residual estimation, and anomaly detection.

Experiments were conducted to evaluate the framework in a collaborative environment where an assembly task is shared between the human and the robot. As a result, the improved trajectory estimations have reduced noise, no discontinuous data, and a median root mean square error (RMSE) of 0.0071 m, about 28 times smaller than that of raw camera-based estimations. Besides, the approach can effectively detect different types of anomalous trajectory changes as soon as 0.624 s after the change onset. We also demonstrate its capability to detect future unknown anomalies and assist in determining a safe working distance. The following are the key contributions:

1) Development of a cost-effective framework for robot trajectory tracking and anomaly detection. It incorporates geometry-based pose estimation, trajectory dynamics, and residual-based statistical process control.
2) The proposed framework overcomes limitations of pose estimation methods associated with external sensors, including inconsistent results, inadequate sensor sampling rate, and low accuracy.
3) It eliminates challenges related to nonlinear profile monitoring in anomaly detection by (i) avoiding an excessive number of monitoring parameters and (ii) capturing variations in the nominal data.

The remainder of the paper is organized as follows: Section II provides a review of related work. Section III elaborates on the experimental details. Section IV presents the proposed framework. Results are presented in Section V and conclusions are given in Section VI.

II. Related work

Existing high-precision external tracking solutions are usually expensive. The absolute positioning accuracy of a calibrated laser tracker can reach approximately 0.0001 to 0.0003 m, but it can cost more than $100,000 [23], [24], let alone installation and software integration. Such high costs hamper the accessibility and scalability of tracking solutions.

To improve the performance of low-cost external sensors, researchers have been studying reference point-based pose estimation. In essence, they attempt to compute the distance between the sensor and some reference points whose coordinates (in certain coordinate systems) are known. For example, a marker-based approach estimates the distance between a camera and a marker by solving the geometric correspondence between marker coordinates and camera-image coordinates via perspective-n-point (PnP) algorithms [25]. These methods have been considered for estimating motion states of various machine components and equipment in manufacturing systems [19], [26]. Although these methods utilize the underlying geometry, they may still generate inconsistent and error-prone results, since they highly depend on environmental conditions (e.g., lighting) and the quality of sensors (e.g., noise level, sampling rate, and accuracy) [17]. Additionally, they can be susceptible to significant computational errors [18].

Alternatively, convolutional neural network (CNN) models have been used to automatically learn the mapping between camera images and the actual poses in a data-driven manner [27]–[33]. While these models may exhibit improved pose estimation accuracy, they may not have captured the underlying geometric principles. Consequently, the accuracy is not guaranteed for future estimations. More importantly, their models lack temporal awareness, and hence may fail to produce sequentially consistent robot poses.

To achieve accurate and consistent tracking of continuous robot poses while incorporating the geometry, efforts have been made to sequentially process the outcomes of reference point-based methods. One common technique is the Kalman filter [19]. However, the Kalman filter and its variations (e.g., extended Kalman filter) have the downside that they assume the robot state at each time step to be Gaussian. Since typical robot trajectories hardly follow a Gaussian distribution, these methods may produce inaccurate results. Moreover, an accurate filtering may require extra measurements of the second order (e.g., velocity and acceleration) [34]. In contrast, recurrent deep learning has facilitated automated calibration of continuous streams of external sensor-based measurements to the ground truth. A recent study demonstrated that LSTM, combined with the Levenberg-Marquardt algorithm, can significantly reduce the camera-based pose estimation error by around three times [35]. Similar enhancements by LSTM have also been observed in the context of other external sensors, such as WiFi devices and accelerometers [36], [37].

On the other hand, even with accurate trajectory measurements, detecting anomalous changes in complex trajectories

presents another major challenge. Conventional process monitoring tools, such as control charts [20], assume the distribution of the monitored variable to be stationary over time, so that they can detect changes when the distribution shifts. Based on this, researchers have explored methods including wavelet transform [38] and polynomial approximation [39] to first characterize the trajectory with stationary parameters and then apply control charts to these parameters. However, this can result in too many parameters to monitor and delays in anomaly detection depending on the resolution of the characterization.

Recent advances in machine learning introduced one-class classification-based anomaly detection techniques that can be easily implemented in near real-time [21], [40]. Nevertheless, they assign all nominal data to the same class, overlooking the variation in nominal data. Consequently, they may have reduced sensitivity in detecting subtle changes (i.e., the early stage of an anomaly) and different types of anomalies.

Residual-based anomaly detection [41], [42] offers several advantages over the aforementioned methods. By focusing on the residuals of the trajectory estimation, which represent the unexplained variation in the data, it excels at capturing unexpected changes and adapts to different changes. Furthermore, the lower-dimensional and more stationary residuals enable straightforward monitoring via control charts [42].

III. Experiment and Data Collection

To establish a meaningful human-robot collaborative environment, the experiment entails an assembly task of a planetary gear system (IRB-approved under protocol IRB2021-0432F). During the collaboration, a UR10 robot arm (Universal Robots, DK) is programmed to sequentially deliver six distinct components to a human operator for assembly. As shown in Fig. 1, the six workpieces (outlined in red dashed boxes) share the same drop-off location (highlighted in green) close to the human. To increase generality and replicate the storage arrangement commonly found in manufacturing settings, the workpieces are initially positioned on fixtures of varying heights. This deliberate choice introduces greater diversity and complexity to the robot trajectory, providing a rich set of dynamic features that can be effectively learned by LSTM models. By successfully verifying our framework's performance on this trajectory, we establish its potential for generalization beyond the current application.

The hardware-level communication between external sensors and internal sensors of the robot are handled by the Robot Operating System (ROS). The trajectory planning problem was resolved by formulating a trajectory that is time-jerk optimal, collision-free, and passes through designated waypoints. Specifically, a nonlinear optimal control problem was constructed and solved via orthogonal collocation [43], [44].

Under nominal conditions, the robot has a fixed trajectory which keeps it at a safe distance from the human operator and workpiece fixtures. Anomalous conditions were designed to be trajectory changes that can potentially cause injury, property damage, and disruption of tasks. However, marginal safety boundaries of the human area and fixtures are defined and enforced in planning algorithms to avoid actual collisions.

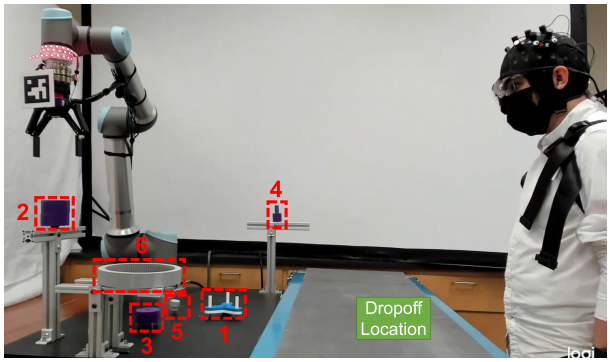

Fig. 1. The initial placement, pickup order, and dropoff location of the six workpieces for the human-robot collaborative assembly task.

As shown in Fig. 2, we introduced three types of trajectory changes (denoted as Type A, B, and C), each to a segment of the trajectory. In Type A change, the robot rushes towards the human while it is supposed to move away for the next workpiece. In Type B change, the robot drops the workpiece at a greater height and near a fixture. During Type C change, the robot approaches the human while it is supposed to go to the drop-off location. The changes were designed to be subtle and rapid to imitate emergency situations. In particular, it takes 1.088 s, 2.896 s, and 1.888 s for the robot to travel from the change onset to the boundary of the human area or fixture in Type A, B, and C changes, respectively. A demonstration video of these designs is included in the Supplementary Material.

In summary, Types A and C correspond to unexpected motions that are potentially harmful to humans. Type B simulates an anomalous change that can cause property damages and task interruptions. For convenience of analysis, an abnormal trajectory only contains one type of change (i.e., one abnormal trajectory segment) and resumes normal operation after the change period instead of shutting down.

In general, we conducted the task 40 times under nominal conditions and 24 times under anomalous conditions (8 times for each type of trajectory change). During each run of the task, we recorded the positions of the robot end-effector (in the robot coordinate system centered at the robot base) using the robot internal sensors at a rate of 125 Hz. These data are considered ground truths for later analyses. An Astra Pro RGB camera with 30 frames per second was employed to implement the marker-based pose estimation. The camera was properly positioned and fixed such that it can remotely and continuously capture images of the marker attached to the end-effector throughout the experiment. The image size is 640×480 and the pose was computed upon receiving each image. The data streams from the pose estimation and internal sensors were synchronized in ROS at a fixed frequency of 125 Hz. Note that both the data from the internal sensors and the pose estimation provide orientation besides location (i.e., 6D pose), but in this study, we focus on the Cartesian coordinates as the approach can be extended to 6D pose.

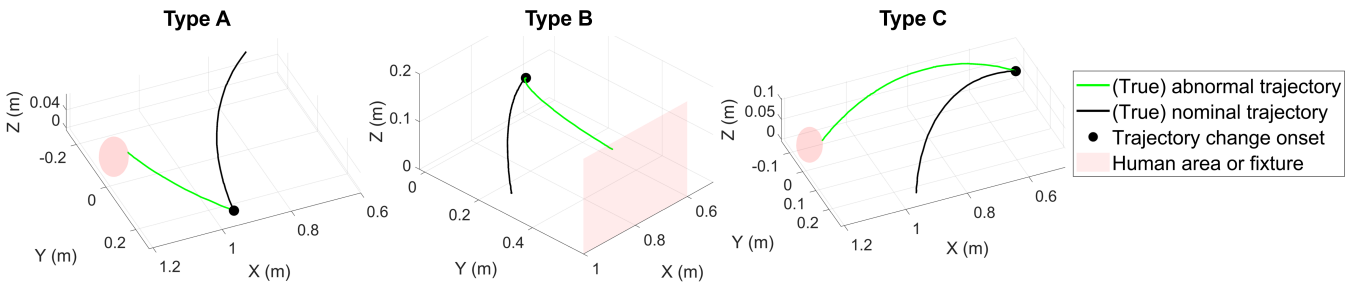

Fig. 2. Illustrations of the three types of trajectory change designs. For each type, it shows the corresponding nominal trajectory segment if the change did not happen. The trajectory segments are the ground truth obtained from the robot's internal sensors.

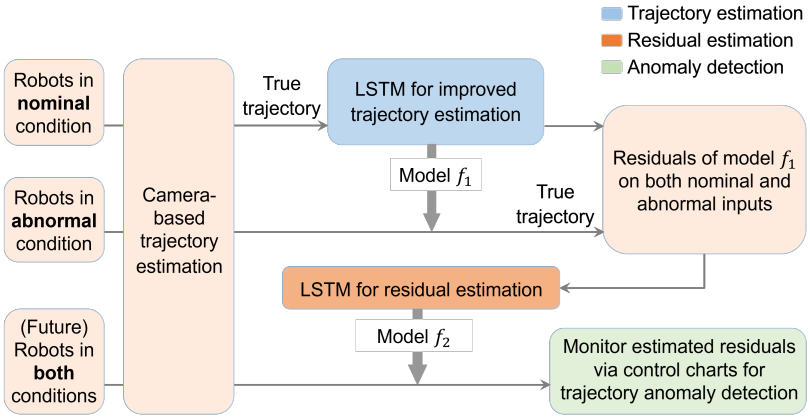

Fig. 3. Workflow of the trajectory estimation, residual estimation, and anomaly detection.

IV. Methodology

Fig. 3 presents the proposed framework consisting of three interconnected threads: (1) estimation of the robot motion trajectory from noisy and inaccurate camera-based pose estimations obtained during normal operation of the robot. Due to the nonlinear, noisy, and transient dynamics underlying the robot motion and the presence of noise in pose estimation, we train an LSTM regression model f_1 for *trajectory estimation*. (2) estimation of model residuals directly from camera-based pose estimation for both nominal and abnormal trajectories. The intuition here is that the accuracy of a well-trained LSTM trajectory estimation model would deteriorate significantly when the robot motion deviates from its nominal trajectories. Considering the nonlinear underlying system dynamics and noisy data, we train another the LSTM regression model f_2 for *residual estimation*. (3) construct a statistical process monitoring scheme for *anomaly detection*, since residuals of trajectory estimation under nominal conditions are expected to follow a stationary distribution that is not necessarily white. The intuition in this thread is that the statistical scheme would signal an alarm whenever the residuals lie outside a certain threshold, which in turn is set to achieve a desired anomaly detection performance.

For *trajectory estimation*, the model f_1 takes a sequence of noisy camera-based pose estimations $\{\widetilde{S}^n_{t-\tau}, ..., \widetilde{S}^n_{t-1}, \widetilde{S}^n_t\}$ and produces as output a more accurate estimation $\widehat{S}^n_t$ at a given time step t. The model f_1 is trained with an aim to minimize the residual, e.g., Euclidean distance between $\widehat{S}^n_t$ and the ground truth S^n_t from internal sensors, at every t. More importantly, f_1 is trained only based on nominal trajectories, and hence it would result in a different residual if the input is from an abnormal trajectory segment $\{\widetilde{S}^a_{t-\tau}, ..., \widetilde{S}^a_{t-1}, \widetilde{S}^a_t\}$. In other words, abnormal residuals ε^a tend to be larger than the nominal residuals ε^n.

For *residual estimation*, the model f_2 takes the same input as f_1 does and outputs an estimate $\widehat{\varepsilon}$ of the corresponding f_1 residual. In this way, only f_2 is needed for future anomaly detection. This enables the framework to work independently without the ground truth from internal sensors and thus provides redundant safety. *Anomaly detection* is realized by monitoring the estimated residuals with a control chart.

A. Camera-based pose estimation

We use camera-based pose estimation to generate input data for trajectory and residual estimation. It uses ArUco markers, a type of binary square fiducial markers that are widely used in the field of computer vision [45]. The goal is to obtain the location of the marker in the camera-centered coordinate system. This is achieved via correspondence between points in a marker-centered coordinate system and their projections in a 2D camera image. There are two advantages to ArUco markers: First, every marker provides four correspondences via its four corners with known coordinates. Second, the unique binary encoding inside each marker makes the method more

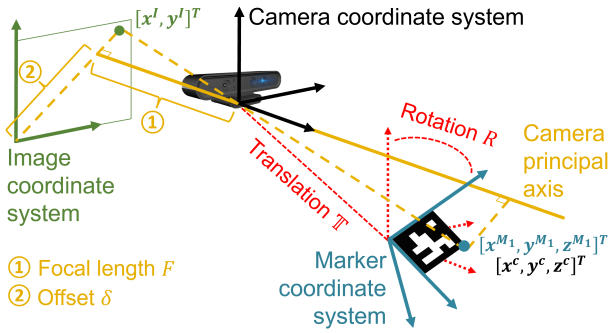

Fig. 4. An illustration of the coordinate transformation of a point from marker coordinates to camera coordinates, then to image coordinates.

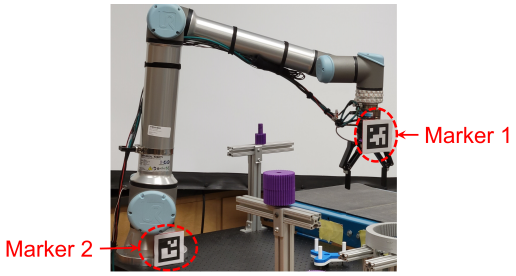

Fig. 5. Placement of the two markers. Marker 1 is attached to the end-effector while Marker 2 is perpendicular and tangent to the robot base.

robust against marker detection error and enables obtaining non-redundant information from different markers.

The aforementioned correspondences can be mathematically expressed as Equation (1), where x^I and y^I are coordinates (pixel indices) in the 2D image along the horizontal and vertical directions, respectively. The offsets along the two directions δ_x and δ_y as well as the focal lengths F_x and F_y are intrinsic parameters of the camera sensor. The rotation matrix $R_1 \in \mathbb{R}^{3\times3}$ and the translation vector $\mathbb{T}_1 \in \mathbb{R}^3$ are the extrinsic parameters and x^{M_1}, y^{M_1}, and z^{M_1} locate a point in the marker coordinate system where the origin is the marker center. The marker coordinates are first transformed into a camera-centered coordinate system based on the extrinsic parameters (highlighted in red in Fig. 4). Then, the camera coordinates are projected onto the image based on the intrinsic parameters (highlighted in yellow in Fig. 4). The coordinates x^I, y^I, x^{M_1}, y^{M_1}, and z^{M_1} can be physically measured by a tape. The parameters F_x, F_y, δ_x, and δ_y can be obtained via camera calibration [46]. Based on the above, R_1 and $\mathbb{T}_1$ can finally be approximated by solving Equation (1) as a PnP problem by using algorithms such as the Efficient PnP [25]. Consequently, we obtain the location of the marker (or end-effector) in the camera-centered coordinate system, denoted as $[x^c, y^c, z^c]^T$.

$$\begin{aligned}\begin{bmatrix} x^I \\ y^I \\ 1 \end{bmatrix} &= \begin{bmatrix} F_x & 0 & \delta_x & 0 \\ 0 & F_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x^c \\ y^c \\ z^c \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} F_x & 0 & \delta_x & 0 \\ 0 & F_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_1 & \mathbb{T}_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x^{M_1} \\ y^{M_1} \\ z^{M_1} \\ 1 \end{bmatrix}\end{aligned} \tag{1}$$

The true end-effector locations provided by the internal robot sensors are in the robot coordinate system with the origin at the center of the robot base. To better compare the pose estimation results with the ground truth, we further computed the marker location in the robot coordinate system, namely $[x^B, y^B, z^B]^T$. This is achieved by attaching another marker (noted as Marker 2) to the robot base (see Fig. 5). It adds two more steps to the coordinate transformation: from the camera coordinate system to Marker 2 coordinate system, and then to the robot coordinate system. The first step is to multiply the previously obtained camera coordinates for Marker 1 by the inverse of the matrix containing extrinsic parameters R_2 and $\mathbb{T}_2$. Both R_2 and $\mathbb{T}_2$ are derived similarly via Equation (1) using the correspondence between points in Marker 2 and their image projections. Finally, we multiply the result by the matrix containing R_3 and $\mathbb{T}_3$ to obtain the location of Marker 1 in the robot coordinate system. Here, R_3 and $\mathbb{T}_3$ can be easily measured by a tape, since Marker 2 is merely perpendicular and tangent to the robot base. The overall coordinate transformation can be written as

$$\begin{bmatrix} x^B \\ y^B \\ z^B \\ 1 \end{bmatrix} = \begin{bmatrix} R_3 & \mathbb{T}_3 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2 & \mathbb{T}_2 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} R_1 & \mathbb{T}_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x^{M_1} \\ y^{M_1} \\ z^{M_1} \\ 1 \end{bmatrix} \tag{2}$$

During the experiment, R_1 and $\mathbb{T}_1$ are updated as the end-effector moves. However, R_2, $\mathbb{T}_2$, R_3, and $\mathbb{T}_3$ are only calculated once prior to the experiment and remain constant, as the positions of the camera, Marker 2, and robot base are fixed.

B. Bidirectional Long Short-Term Memory (Bi-LSTM) [22]

We use Bi-LSTM models for both trajectory and residual estimation in our framework. As mentioned in Section I, camera-based pose estimation may capture general trajectory patterns, but it still suffers from high noise, insufficient sampling rate, and low accuracy. This is possibly due to fast robot motion, a low camera resolution and frame rate, computational errors, etc. Therefore, an LSTM is employed to improve the large amount of poor-quality pose estimates. Moreover, to make the method independent of internal sensors, we build another LSTM to estimate residuals of the first LSTM model based on the same input.

To formalize, our goal is to get the location estimation $\widehat{S}_t \in \mathbb{R}^3$ or the residual estimation $\widehat{\varepsilon}_t \in \mathbb{R}$ at time t. This is achieved by learning the mapping between a series of camera-based pose estimation $\{\widetilde{S}_{t-\tau}, ..., \widetilde{S}_{t-1}, \widetilde{S}_t\}$ (wherein $\widetilde{S}_t = [x_t^B, y_t^B, z_t^B]^T$) and the true location $S_t = [x_t, y_t, z_t]^T$ or the true residual ε_t, respectively. As implied by its name, an LSTM captures both long- and short-term temporal patterns in sequential data, enabling it to effectively learn the mapping.

Bi-LSTM is a type of LSTM model that processes an input sequence in a two-direction fashion, i.e., from time step $t-\tau$ to t and from t to $t-\tau$. This is realized using the so-called forward and backward cells in each Bi-LSTM layer

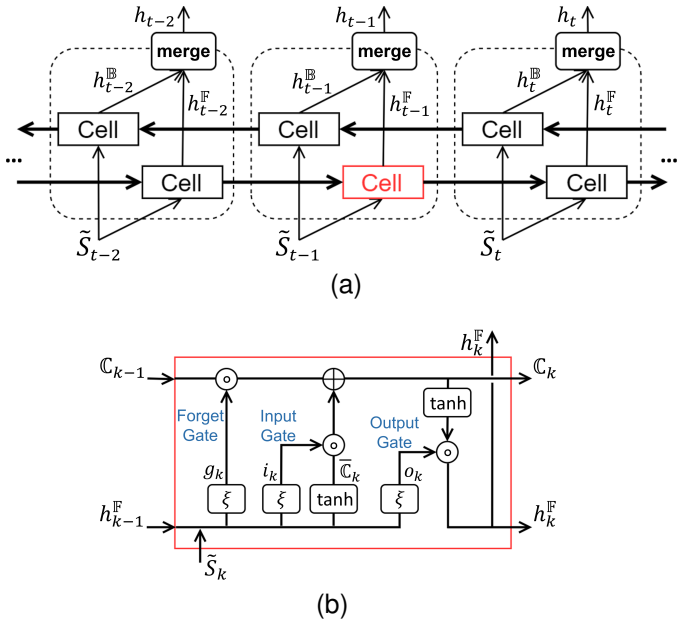

Fig. 6. (a) A snippet of one layer of the Bi-LSTM model which processes the camera-based pose estimation at time steps $t-2$, $t-1$, and t. It contains two sets of LSTM cells, one of which processes the data forward and the other processes the data backward. (b) Operations within an LSTM cell at the time step k.

of the model (see Fig. 6a). The number of cells along each direction is equal to the number of time steps in the input. A cell takes information from the preceding cell as well as the corresponding time step of the input. Next, the outputs (also called hidden states) from the forward cell (denoted as $h_{\cdot}^{\mathbb{F}} \in \mathbb{R}^{d_j}$) and the backward cell (denoted as $h_{\cdot}^{\mathbb{B}} \in \mathbb{R}^{d_j}$) are merged into one output $h_{\cdot} \in \mathbb{R}^{2d_j}$. Here, d_j is the user-specified hidden state dimension in the j^{th} layer. Then, all the $h_{\cdot}$ form a new sequence $\{h_{t-\tau}, ..., h_t\}$ which is fed into the next layer. Finally, the model uses a weight $w_D \in \mathbb{R}^{2d_j \times 3}$ or $w_D \in \mathbb{R}^{2d_j}$ to map h_t from the last Bi-LSTM layer onto the final output $\widehat{S}_t$ or $\widehat{\varepsilon}_t$, respectively.

Fig. 6b and Equations (3) elaborate the operations carried out in a forward cell at time k. The forget gate determines the proportion $g_k \in [0,1]^{d_j}$ of the previous cell state $\mathbb{C}_{k-1} \in \mathbb{R}^{d_j}$ to retain in the current cell state $\mathbb{C}_k$. The input gate decides the portion i_k of new information (also called a candidate $\overline{\mathbb{C}}_k$) to incorporate into $\mathbb{C}_k$. The output gate determines the proportion o_k of $\tanh(\mathbb{C}_k)$ to output as the current hidden state $h_k^{\mathbb{F}}$. Here, for convenience, $\tanh(\mathbb{C}_k)$ denotes an element-wise operation that converts each element of $\mathbb{C}_k$ into a value between -1 and 1. This captures data nonlinearity and controls the magnitude and sign of the hidden state.

To obtain g_k, i_k, and o_k, the previous hidden state h_{k-1} and the current input $\widetilde{S}_k$ are each multiplied by a weight and then added together with a bias vector. Finally, a sigmoid function $\xi()$ is applied to give each element of g_k, i_k, and o_k a value between 0 and 1. The candidate $\overline{\mathbb{C}}_k$ is obtained similarly except that $\xi()$ is replaced by $\tanh()$ so that $\overline{\mathbb{C}}_k$ can either positively or negatively contribute to the current cell state $\mathbb{C}_k$. All the weights and biases are initialized with random values. During the model training, they are updated iteratively to minimize the error between the model output and the ground truth.

$$g_k = \xi(w_{gh} h_{k-1}^{\mathbb{F}} + w_{gS} \widetilde{S}_k + b_g) \tag{3a}$$
$$i_k = \xi(w_{ih} h_{k-1}^{\mathbb{F}} + w_{iS} \widetilde{S}_k + b_i) \tag{3b}$$
$$o_k = \xi(w_{oh} h_{k-1}^{\mathbb{F}} + w_{oS} \widetilde{S}_k + b_o) \tag{3c}$$
$$\overline{\mathbb{C}}_k = \tanh(w_{\mathbb{C}h} h_{k-1}^{\mathbb{F}} + w_{\mathbb{C}S} \widetilde{S}_k + b_{\mathbb{C}}) \tag{3d}$$
$$\mathbb{C}_k = g_k \circ \mathbb{C}_{k-1} + i_k \circ \overline{\mathbb{C}}_k \tag{3e}$$
$$h_k^{\mathbb{F}} = o_k \circ \tanh(\mathbb{C}_k) \tag{3f}$$

wherein $w_{\cdot h} \in \mathbb{R}^{d_j \times d_j}$, $w_{\cdot S} \in \mathbb{R}^{d_j \times 3}$. $\circ$ is element-wise multiplication, $b_{\cdot}$ is the bias vector. For any $q \in \mathbb{R}$, $\xi(q) = 1/(1+e^{-q})$ and $\tanh(q) = (e^{2q}-1)/(e^{2q}+1)$.

C. Residual control chart

We use Shewhart control charts for anomaly detection in our framework. A Shewhart control chart [20] is a statistical process control tool widely used to determine if a process is in control or out of control. It sets two thresholds, namely an upper control limit (UCL) and a lower control limit (LCL), to detect anomalies that fall out of them over time. In this study, we employ a Shewhart control chart to monitor the output of f_2 (i.e., estimated residuals $\widehat{\varepsilon}$) during a trajectory. This is because estimated residuals of nominal trajectory segments are much more stationary and smaller than that of abnormal trajectory segments (elaborated in Section V-B). For convenience, we name these estimated residuals as nominal estimated residuals and abnormal estimated residuals in the later context.

The procedure is divided into Phase I and Phase II. In Phase I, we establish UCL and LCL based on the mean μ_0 and standard deviation σ of nominal estimated residuals, that is

$$\text{UCL} = \mu_0 + \sigma \cdot \Phi^{-1}(1-\alpha/2) \tag{4a}$$
$$\text{LCL} = \mu_0 - \sigma \cdot \Phi^{-1}(1-\alpha/2) \tag{4b}$$

wherein $\Phi^{-1}(1-\alpha/2)$ returns the $(1-\alpha/2)$-percentile of a standard Gaussian distribution. Here, α is determined such that the control limits include as many nominal estimated residuals as possible, while excluding (or detecting) an abnormal estimated residual as soon as possible. In Phase II, we evaluate the test performance of the established control chart by applying it to the estimated residuals of newly observed trajectories.

Two types of average running lengths (denoted as ARL_0 and ARL_1) are commonly used to assess the statistical performance of control charts. ARL_0 measures the average number of consecutive observations within control limits when the process is actually in control. It is computed as

$$\text{ARL}_0 = \frac{1}{\alpha} \tag{5}$$

ARL_1 measures the average delay in detecting a mean shift as the average number of consecutive observations within control limits after the process mean μ_0 shifts to μ_1, that is

$$\text{ARL}_1 = \frac{1}{1 - \Phi\big(\Phi^{-1}(1-\frac{\alpha}{2}) - \rho\big) + \Phi\big(-\Phi^{-1}(1-\frac{\alpha}{2}) - \rho\big)} \tag{6}$$

wherein $\rho = |\mu_1 - \mu_0|/\sigma$. It is important to note that even when the data is non-Gaussian, ARL_0 and ARL_1 still provide a reference for evaluating the performance of a control chart.

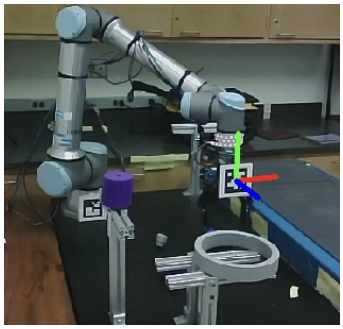

Fig. 7. An RGB image capturing Marker 1, with its coordinate system identified, from the camera during the experiment.

V. Results

A. LSTM-enhanced trajectory estimation

Prior to the experiment, the camera was calibrated using Zhang's method [46], which involved capturing images of a chessboard pattern. This process allowed us to obtain intrinsic parameters of the camera, i.e., offsets and focal lengths. In addition, R_2, $\mathbb{T}_2$, R_3, and $\mathbb{T}_3$ were calculated based on Marker 2. During the experiment, every frame captured by the camera underwent thresholding and contour analysis [45] to automatically identify Marker 1 and its coordinate system (see Fig. 7 for an example). Then, the pose of Marker 1 in the robot coordinate system was estimated via Equation (2).

Fig. 8a shows the results of camera-based raw pose estimation. The two estimated nominal trajectories from the same experiment have large noise and are inconsistent with each other. They also fail to accurately match the true trajectory with an RMSE of 0.1947 m. Here, RMSE is computed by $(\sum_t^T ||S_t - \widetilde{S}_t||_2^2/T)^{1/2}$, wherein T is the total number of time steps in a trajectory. Furthermore, Fig. 8c suggests that although camera-based raw estimations capture the general trajectory pattern, there are many discontinuous measurements. These drawbacks are possibly caused by computational errors in extrinsic parameters as well as poor marker detections due to bad lightening, fast robot motion, low camera resolution, and low camera frame rate.

With the use of Bi-LSTM, the nominal trajectory estimation can be significantly improved in terms of accuracy and noise level. As demonstrated in Fig. 8b, a Bi-LSTM estimation yielded an RMSE of 0.0068 m, which is about 28 times smaller than that of the camera-based raw estimations. It also "improves" the sampling rate of the external sensor (i.e., the camera) by interpolating all the discontinuous data points with accurate estimations. These enhancements collectively contribute to a more precise anomaly detection.

The model f_1 has two Bi-LSTM layers, with hidden state dimensions $d_1 = 30$ and $d_2 = 15$ in the first and the second layer, respectively. The model was trained and selected via a ten-fold cross validation using camera-based estimations from 34 nominal trajectories. The training process spanned 1000 epochs with a learning rate of 2×10^{-4}, during which the model converged to a local minimum of residuals. Then, the model was tested on 6 nominal trajectories and 24 abnormal

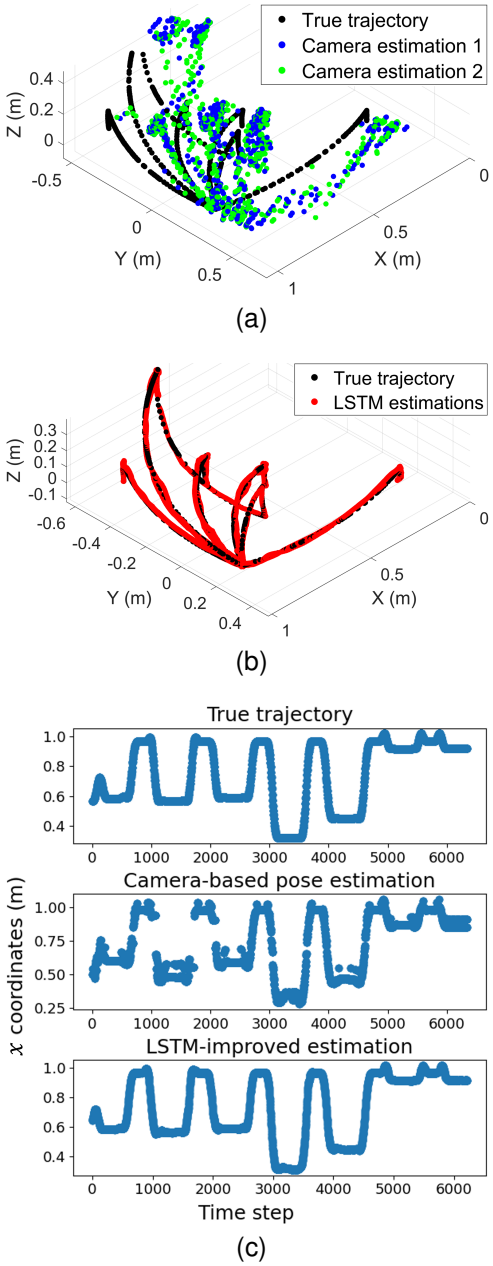

Fig. 8. (a) The 3D perspective of the true trajectory and two camera-based trajectory estimations with an RMSE of 0.1947 m. (b) An example of the LSTM-improved trajectory estimation with a test RMSE of 0.0068 m. (c) x coordinates from the ground truth, the camera-based estimation, and the LSTM-improved estimation.

trajectories. In each trajectory, the input sequences were created by using a sliding window of length 125 (i.e., τ = 124) and a step size of 1. In this way, after the first second of the trajectory, the model can produce estimations at 125 Hz, which is the same as the internal sensor sampling rate. Overall, the training data contain 211,693 input sequences and the test data have 185,798 input sequences.

Fig. 9 summarizes the performance of f_1 on the test data from abnormal and nominal trajectories, respectively. It is clear that abnormal trajectories always have a larger RMSE than the nominal ones. In particular, the median RMSE of abnormal trajectories (0.0366 m) is as 5 times larger as that of the

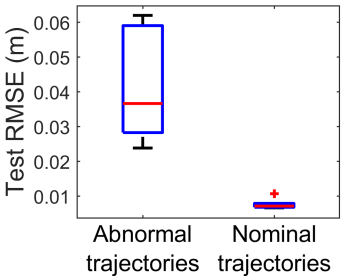

Fig. 9. Test RMSE distribution of the LSTM model regarding nominal and abnormal trajectories.

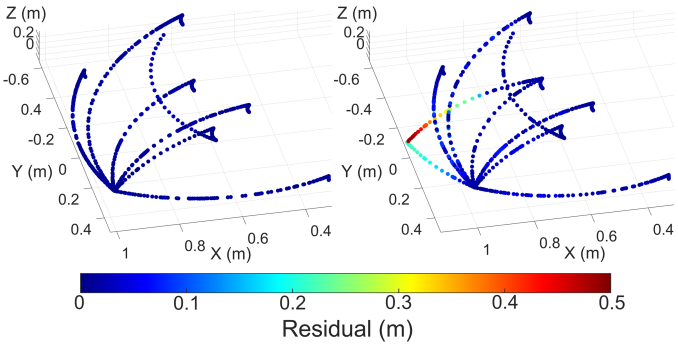

Fig. 10. f_1 residuals (in color coding) at each time step of a nominal (left) and an abnormal (right) trajectory from the test data.

nominal ones (0.0071 m). Such a noticeable distinction implies potential opportunities for detecting abnormal trajectory segments based on residual patterns of f_1.

B. Trajectory change detection

We further study how residuals of f_1 (i.e., Euclidean distance between the model output $\widehat{S}_t$ and the true coordinates S_t) are distributed in a trajectory. Specifically, we calculated the residual at each time step. As seen in Fig. 10, the residuals are distinctively larger only during the trajectory change while being much smaller and more stationary in nominal trajectory segments. Therefore, large residuals can serve as an indicator of anomalous trajectory changes during continuous monitoring. This makes the anomaly detection simpler and more efficient compared to directly monitoring the nonlinear robot trajectory. However, in situations where internal sensors malfunction or are under cyber attacks, the true locations become unavailable and so do the residuals. To make the anomaly detection independent of internal sensors and provide redundant safety, we train another Bi-LSTM model f_2 to estimate the residual of f_1.

1) Residual estimation: Model f_2 has two Bi-LSTM layers, with hidden state dimensions $d_1 = 12$ in the first layer and $d_2 = 8$ in the second layer. Similar to f_1, the inputs to f_2 are sequences of camera-based estimations with the same sliding window length $\tau + 1$ and step size 1. f_2 was trained over 600 epochs on a dataset comprising 211,706 sequences from 34 nominal trajectories and 2,869 sequences from 8 abnormal trajectories (4 for Type A and 4 for Type B). Here,

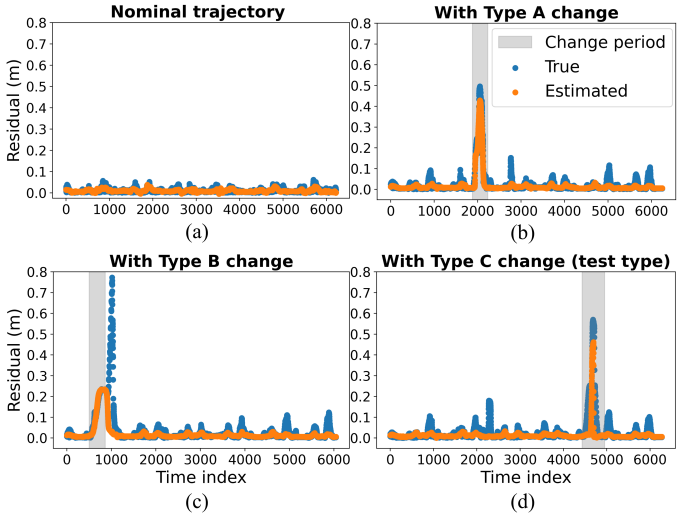

Fig. 11. Estimated and true residuals for (a) a nominal test trajectory, (b) a Type A abnormal test trajectory, (c) a Type B abnormal test trajectory, and (d) a completely unknown Type C abnormal trajectory.

the abnormal sequences were created using only trajectory segments that cover anomalous changes. Next, f_2 was tested on 37,366 sequences from six nominal trajectories and 6,986 sequences from 16 abnormal ones (4 for Type A, 4 for Type B, and 8 for Type C). It is important to note that the training abnormal data only include Type A and B changes, but the test abnormal data include all three types. This blind test assesses the approach's generalization performance on future unknown types of trajectory changes.

The model has a training RMSE of 0.0094 m and a test RMSE of 0.0494 m. Fig. 11 shows the residual estimation results of the model tested on a nominal trajectory and on A-, B-, and C-Type abnormal trajectories. Although f_2 may not provide an exact estimation of the residual magnitude, it clearly identifies the residual difference between nominal and abnormal inputs. This is possibly owing to the fact that f_1 has distinctively larger residuals on abnormal inputs. More importantly, f_2 also captures the propensity of the residuals transitioning from nominal to abnormal. This allows us to detect an anomalous change at its early stage, even when the change happens rapidly in a short period of time (see Fig. 11(b)-(d)). Besides, the test result on Type C abnormal trajectories indicates that the model and the pattern of estimated residuals can generalize well on future unknown types of anomalous changes.

2) Control chart anomaly detection: In Phase I, we set UCL and LCL by determining the value of α based on estimated residuals of the training data of f_2, including 34 nominal and 8 abnormal trajectories. Specifically, α is determined so that the control chart does not alarm on the nominal estimated residuals, but raises an alarm on the abnormal estimated residuals as soon as possible. In the training data, the mean μ_0 and standard deviation σ of nominal estimated resiudals are 0.0096 m and 0.006 m, respectively. As a result, α was set as 1×10^{-8} and all the nominal estimated residuals fell between the LCL (-0.0247 m) and the UCL (0.0439 m).

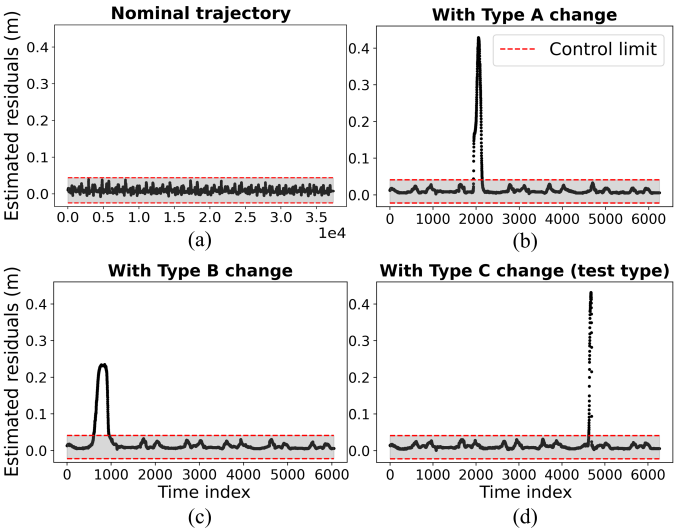

Fig. 12. Phase II monitoring of the estimated residuals for (a) all test nominal trajectories, (b) a test Type A, (c) a test Type B, and (d) a completely unknown Type C abnormal trajectory.

Regarding abnormal training data, the control chart can detect Type A trajectory changes (indicated by the estimated residuals that exceed the control limits for the first time) in an average of 80 $\pm$ 2.773 time steps after the onset of the change. This is around 0.64 $\pm$ 0.022 s at a sampling rate of 125 Hz. The average detection time for Type B changes is 133 $\pm$ 2.487 time steps (around 1.064 $\pm$ 0.02 s) after the onset of the change.

In Phase II, the established control chart was employed to monitor estimated residuals of the test data of f_2, including 6 nominal and 16 abnormal trajectories. Fig. 12(a) shows that all estimated residuals for the 6 test nominal trajectories fell within the control limits, i.e., no false alarms. Besides, the chart can successfully detect unseen Type A trajectory changes after an average of 84 $\pm$ 6.042 time steps (around 0.672 $\pm$ 0.048 s) past the change onset. An unseen Type B trajectory change can be detected on average at 135 $\pm$ 1.658 time steps (around 1.08 $\pm$ 0.013 s). Moreover, it took the chart 207 $\pm$ 1.833 time steps (around 1.656 $\pm$ 0.015 s) to detect a completely unknown trajectory change (Type C). Fig. 12(b)-(d) present the cases with the shortest detection time (0.624 s, 1.056 s, and 1.624 s) for Type A, B, and C changes, respectively. Notably, according to the Transmission Control Protocol/Internet Protocol (TCP/IP), the data transmission latency of the sensor is usually in the order of milliseconds, which is negligible compared to the magnitude of the detection time [4].

Table I provides a comparison between the detection performance of the Shewhart chart and other commonly used control charts, including the cumulative sum (CUSUM) [47], exponentially weighted moving average (EWMA) [47], and Hotelling T^2 charts [48]. Here, CUSUM and EWMA charts were evaluated on the same estimated residuals as the Shewhart chart. However, the construction of a Hotelling T^2 multivariate chart involved modifying the output dimension of f_2 from 1 to 3, thereby generating separate residual estimations along the x, y, and z directions. During Phase I, the parameters of the CUSUM and EWMA charts were tuned so that all nominal estimated residuals of the training data fell exactly within the control limits. On the other hand, since the Hotelling T^2 chart shares the same parameter α with the Shewhart chart, we set $\alpha = 10^{-8}$ for both charts. As a result, in Phase II, no false alarms were triggered by the Shewhart, CUSUM, and EWMA charts. In contrast, the Hotelling T^2 chart produced false alarms in 3 out of the 6 test nominal trajectories. These false alarms were potentially caused by the Hotelling T^2 chart's heightened sensitivity to mean shifts in a multivariate Gaussian distribution, where the impact of the nominal estimated residuals slightly deviating from a Gaussian distribution might be amplified. In terms of detecting anomalies in the 16 test abnormal trajectories, the Hotelling T^2 chart managed to detect Type A change faster than the Shewhart chart. However, it again had false alarms prior to the actual detection. The Shewhart chart also demonstrated superior performance compared to the CUSUM and EWMA charts. This is possibly because the sensitivity of the CUSUM and EWMA charts were reduced by tuning the parameters in Phase I to prevent false alarms.

Fig. 13 further illustrates the best Phase II detection results in Fig. 12(b)-(d) from a real-world perspective. Our approach successfully detected all three types of anomalous changes before the robot reaches the boundary of human area or fixture (i.e., the designated collision point). Specifically, the Type A change was detected 0.1467 m away from and 0.464 s prior to the designated collision point. For the Type B change, the detection point is 0.2368 m and 1.84 s before the collision point. Even in the case of an unknown Type C change, our approach provides a safety buffer of 0.0322 m and 0.264 s. These detection results enable collision prevention strategies such as a prompt automatic shutdown mechanism.

We assess the statistical performance of the control chart using metrics such as ARL_0 and ARL_1. With $\alpha = 1 \times 10^{-8}$, the control chart has an ARL_0 of 1×10^8 based on Equation (5). This indicates that the control chart will not make a false alarm under the nominal condition within an average of 1×10^8 time steps. On the other hand, the ARL_1 is 4, 24, 316, 10475, and 893987 time steps for detecting a mean shift of magnitude 5σ, 4σ, 3σ, 2σ, and σ, respectively. This provides an idea of how sensitive our approach is to the anomalous change in terms of detection time. For example, if the nominal estimated residuals are Gaussian, our approach can detect abnormal estimated residuals that exceed $\mu_0 + 4\sigma$ within an average of 24 times steps (approximately 0.192 s). It becomes less sensitive to smaller mean shifts.

We also evaluate the sensitivity of our approach in terms of the magnitude of deviations from the nominal trajectory. Such information, in turn, enables us to establish a safe working distance between the robot and the human. As designed in Fig. 2, the magnitude of all types of deviations are monotonically increasing over time. This allows us to measure the sensitivity by the distance between the detection point and its parallel point in the nominal trajectory, i.e., its temporally equivalent point if the anomalous change had not happen (marked as black stars in Fig. 13). As a result, the distance between the detection point and the parallel point ranges from 0.1796 m

TABLE I
CHANGE DETECTION PERFORMANCE COMPARISON

Method	Number of false alarms out of 6 test nominal trajectories	Detection time in 16 test abnormal trajectories		
		Type A	Type B	Type C
CUSUM	0	0.784 $\pm$ 0.048 s	1.184 $\pm$ 0.019 s	1.754 $\pm$ 0.019 s
EWMA	0	0.69 $\pm$ 0.049 s	1.092 $\pm$ 0.013 s	1.659 $\pm$ 0.016 s
Hotelling T^2	3	0.626 $\pm$ 0.014 s (with false alarms)	1.208 $\pm$ 0.033 s	1.728 $\pm$ 0.022 s (with false alarms)
Shewhart	0	**0.672** $\pm$ **0.048** s	**1.08** $\pm$ **0.013** s	**1.656** $\pm$ **0.015** s

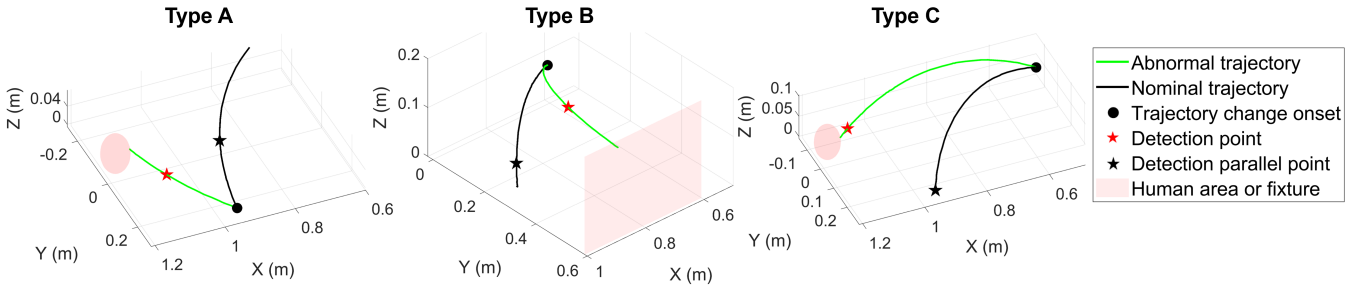

Fig. 13. Real-world perspective of the anomaly detection results regarding the three types of trajectory change, corresponding to Fig. 12(b)-(d).

to 0.3697 m among all the Phase II detection results (0.21 $\pm$ 0.0377 m, 0.1864 $\pm$ 0.0018 m, and 0.3692 $\pm$ 0.0004 m for Type A, B, and C changes, respectively). In our case, the safe working distance can be conservatively set at the maximum distance of 0.3697 m to account for the worst-case scenario.

VI. CONCLUSIONS

We proposed a cost-efficient framework for tracking the trajectory of a collaborative robot during its nominal operations as well as for detecting anomalous trajectory changes. By leveraging an external IoT sensor (e.g., an RGB camera), the framework works remotely and independently of the robot system. This enhances resilience against cyber attacks and provides redundant safety measures against inherent robot defects.

The framework addresses limitations of external sensors by employing LSTM models that capture complex temporal patterns in sensor measurements and provide better measurements. Specifically, it reduces noise in the data, interpolates discontinuous data, and improves measurement accuracy. It also addresses challenges related to nonlinear profile monitoring in anomaly detection (e.g., an excessive number of parameters to monitor and limited generalizability on future unknown trajectory changes) by utilizing residual control charts. The framework can be extended to tracking and monitoring other objects. It can also be applied to other kinds of external sensors, such as 3-axis accelerometers and gyroscopes, to enhance their measurements and to detect anomalies.

Experiments were conducted to examine the performance and practicality of the framework in a real-world human-robot collaborative environment. As a result, it estimated the robot trajectory with a median RMSE of 0.0071 m, about 28 times more accurate than that of the camera-based raw estimations. Besides, it successfully detected various types of anomalous trajectory changes as fast as 0.624 s after the change onset. The detection was also shown to work on unknown anomalous changes. Overall, residual anomalies that have a mean shift over 4σ can be detected within an average of 0.192 s according to ARL_1. The framework also assisted in determining a safe working distance of 0.3697 m, which represents the worst-case motion deviation upon detecting an anomaly.

Future work that evaluates this framework against thresholds of human perceptions and actions could be valuable to (1) inform bioinspired framework refinement, or (2) determine under what scenarios would the framework (using external sensors) or human perceptions be more effective in detecting anomalous robot motion. There are also certain limitations that can be addressed in the future. Firstly, the framework can be expanded to monitor other robot joints. This can be achieved by incorporating additional markers, taking advantage of the scalability offered by the cost-efficient framework. Secondly, one can monitor 6D robot poses by changing the input and output dimensions of the LSTM models.

REFERENCES

[1] L. Liu, F. Guo, Z. Zou, and V. G. Duffy, "Application, development and future opportunities of collaborative robots (cobots) in manufacturing: A literature review," *Int. J. Hum.-Comput. Interact.*, pp. *1–18*, 2022.

[2] D. Romero, P. Bernus, O. Noran, J. Stahre, and Å. Fast-Berglund, "The operator 4.0: Human cyber-physical systems & adaptive automation towards human-automation symbiosis work systems," in *IFIP WG 5.7 Int. Conf. Adv. Prod. Manag. Syst.*, Iguassu Falls, Brazil, 2016, pp. *677–686*.

[3] A. Hanchate, P. S. Dave, A. Tiwari, D. Sagapuram, A. Verma, S. R. Kumara, and S. T. Bukkapatnam, "A graphical representation of sensor mapping for machine tool fault monitoring and prognostics for smart manufacturing," *Smart Sustain. Manuf. Syst.*, vol. 7, no. 1, pp. *82–110*, 2023.

[4] H. Yang, S. Kumara, S. T. Bukkapatnam, and F. Tsung, "The internet of things for smart manufacturing: A review," *IISE Trans.*, vol. 51, no. 11, pp. *1190–1216*, 2019.

[5] S. T. Bukkapatnam, K. Afrin, D. Dave, and S. R. Kumara, "Machine learning and AI for long-term fault prognosis in complex manufacturing systems," *CIRP Annals*, vol. 68, no. 1, pp. *459–462*, 2019.

[6] Z. M. Bi, C. Luo, Z. Miao, B. Zhang, W. J. Zhang, and L. Wang, "Safety assurance mechanisms of collaborative robotic systems in manufacturing," *Robot. Comput.-Integr. Manuf.*, vol. 67, 2021, Art. no. *102022*.

[7] B. Dieber, B. Breiling, S. Taurer, S. Kacianka, S. Rass, and P. Schartner, "Security for the robot operating system," *Robot. Auton. Syst.*, vol. 98, pp. *192–203*, 2017.

[8] S. Bharti and A. McGibney, "CoRoL: a reliable framework for computation offloading in collaborative robots," *IEEE Internet Things J.*, vol. 9, no. 19, pp. *18195–18207*, 2022.

[9] S. Kianoush, S. Savazzi, M. Beschi, S. Sigg, and V. Rampa, "A multisensory edge-cloud platform for opportunistic radio sensing in cobot environments," *IEEE Internet Things J.*, vol. 8, no. 2, pp. *1154–1168*, 2020.

[10] P. Wellener, S. Shepley, B. Dollar, S. Laaper, H. A. Manolian, and D. Beckoff, "2019 Deloitte and MAPI smart factory study," Deloitte, London, UK, 2019. [Online] Available: https://www2.deloitte.com/content/dam/insights/us/articles/6276_2019-Deloitte-and-MAPI-Smart-Factory-Study/DI_2019-Deloitte-and-MAPI-Smart-Factory-Study.pdf

[11] N. Gupta, A. Tiwari, S. T. Bukkapatnam, and R. Karri, "Additive manufacturing cyber-physical system: Supply chain cybersecurity and risks," *IEEE Access*, vol. 8, pp. *47322–47333*, 2020.

[12] P. Mahesh, A. Tiwari, C. Jin, P. R. Kumar, A. N. Reddy, S. T. Bukkapatanam, N. Gupta, and R. Karri, "A survey of cybersecurity of digital manufacturing," *Proc. IEEE*, vol. 109, no. 4, pp. *495–516*, 2020.

[13] K. S. Fu, R. C. Gonzalez, and C. G. Lee, "Sensing," in *Robotics*. New York, USA: MHHE, 1983, ch. 6, pp. *267–295*.

[14] A. Chowdhury, G. Karmakar, and J. Kamruzzaman, "Survey of recent cyber security attacks on robotic systems and their mitigation approaches," in *Detecting and Mitigating Robotic Cyber Security Risks*. PA, USA: IGI global, 2017, ch. 19, pp. *284–299.*

[15] W. R. Dunn, "Designing safety-critical computer systems," *Computer*, vol. 36, no. 11, pp. *40–46*, 2003.

[16] L. Xing, "Reliability in Internet of Things: Current status and future perspectives," *IEEE Internet Things J.*, vol. 7, no. 8, pp. *6704–6721*, 2020.

[17] M. B. Alatise and G. P. Hancke, "Pose estimation of a mobile robot based on fusion of IMU data and vision data using an extended Kalman filter," *Sensors*, vol. 17, no.10, 2017, Art. no. *2164*.

[18] F. von Drigalski, K. Hayashi, Y. Huang, R. Yonetani, M. Hamaya, K. Tanaka, and Y. Ijiri, "Precise multi-modal in-hand pose estimation using low-precision sensors for robotic assembly," in *Proc. IEEE Int. Conf. Robot. Autom.*, Xi'an, China, 2021, pp. *968–974*.

[19] Y. Zhong, Z. Wang, A. V. Yalamanchili, A. Yadav, B. R. Srivatsa, S. Saripalli, and S. T. Bukkapatnam, "Image-based flight control of unmanned aerial vehicles (UAVs) for material handling in custom manufacturing," *J. Manuf. Syst.*, vol. 56, pp. *615–621*, 2020.

[20] W. A. Jensen, L. A. Jones-Farmer, C. W. Champ, and W. H. Woodall, "Effects of parameter estimation on control chart properties: a literature review," *J. Qual. Technol.*, vol. 38, no. 4, pp. *349–364*, 2006.

[21] L. Shen, Z. Li, and J. Kwok, "Timeseries anomaly detection using temporal hierarchical one-class network," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. *13016–13026*, 2020.

[22] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. *2673–2681*, 1997.

[23] A. Nubiola and I. A. Bonev, "Absolute calibration of an ABB IRB 1600 robot using a laser tracker," *Robot. Comput.-Integr. Manuf.*, vol. 29, no. 1, pp. *236–245*, 2013.

[24] Y. Liu, Y. Li, Z. Zhuang, and T. Song, "Improvement of robot accuracy with an optical tracking system," *Sensors*, vol. 20, no. 21, 2020, Art. no. *6341*.

[25] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate $O(n)$ solution to the PnP problem," *Int. J. Comput. Vis.*, vol. 81, pp. *155–166*, 2009.

[26] Z. Wang, Z. Wang, W. H. Ko, A. S. Iquebal, V. Nguyen, N. A. Kazerooni, Q. Ma, A. Srinivasa, P. R. Kumar, and S. Bukkapatnam, "An autonomous laser kirigami method with low-cost real-time vision-based surface deformation feedback system," *Int. J. Adv. Manuf. Technol.*, vol. 118, pp. *1873–1883*, 2022.

[27] Y. Hu, P. Fua, W. Wang, and M. Salzmann, "Single-stage 6D object pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Seattle, WA, USA, 2020, pp. *2930–2939*.

[28] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DoF camera relocalization," in *Proc. IEEE Int. Conf. Comput. Vis.*, Santiago, Chile, 2015, pp. *2938–2946*.

[29] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," *Robot.: Sci. Syst.*, vol. 43, no. 12, pp. *41–49*, 2017.

[30] G. Wang, F. Manhardt, F. Tombari, and X. Ji, "GDR-Net: Geometry-guided direct regression network for monocular 6D object pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Nashville, TN, USA, 2021, pp. *16611–16621*.

[31] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise voting network for 6-DoF pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Long Beach, CA, USA, 2019, pp. *4561–4570*.

[32] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield, "Camera-to-robot pose estimation from a single image," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. *9426–9432*.

[33] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. *5974–5983*.

[34] H. Moradkhani, S. Sorooshian, H. V. Gupta, and P. R. Houser, "Dual state–parameter estimation of hydrological models using ensemble Kalman filter," *Adv. Water Resour.*, vol. 28, no. 2, pp. *135–147*, 2005.

[35] D. K. Bilal, M. Unel, and L. T. Tunc, "Improving vision based pose estimation using LSTM neural networks," in *Proc. Annu. Conf. IEEE Ind. Electron. Soc.*, 2020, pp. *483–488*.

[36] M. Zhang, J. Jia, J. Chen, Y. Deng, X. Wang, and A. H. Aghvami, "Indoor localization fusing WiFi with smartphone inertial sensors using LSTM networks," *IEEE Internet Things J.*, vol. 8, no. 17, pp. *13608–13623*, 2021.

[37] Q. Wang, L. Ye, H. Luo, A. Men, F. Zhao, and Y. Huang, "Pedestrian stride-length estimation based on LSTM and denoising autoencoders," *Sensors*, vol. 19, no. 4, 2019, Art. no. *840*.

[38] K. Paynabar and J. Jin, "Characterization of non-linear profiles variations using mixed-effect models and wavelets," *IIE Trans.*, vol. 43, no. 4, pp. *275–290*, 2011.

[39] S. I. Chang and S. Yadama, "Statistical process control for monitoring non-linear profiles using wavelet filtering and B-spline approximation," *Int. J. Prod. Res.*, vol. 48, no. 4, pp. *1049–1068*, 2010.

[40] Y. Feng, J. Chen, Z. Liu, H. Lv, and J. Wang, "Full graph autoencoder for one-class group anomaly detection of IIoT system," *IEEE Internet Things J.*, vol. 9, no. 21, pp. *21886–21898*, 2022.

[41] Y. Wang, X. Du, Z. Lu, Q. Duan, and J. Wu, "Improved LSTM-based time-series anomaly detection in rail transit operation environments," *IEEE Trans. Ind. Inform.*, vol. 18, no. 12, pp. *9027–9036*, 2022.

[42] F. Kadri, F. Harrou, S. Chaabane, Y. Sun, and C. Tahon, "Seasonal ARMA-based SPC charts for anomaly detection: Application to emergency department systems," *Neurocomputing*, vol. 173, pp. *2102–2114*, 2016.

[43] Y. Wen and P. Pagilla, "Path-constrained and collision-free optimal trajectory planning for robot manipulators," *IEEE Trans. Autom. Sci. Eng.*, pp. *763–774*, 2022.

[44] Y. Wen and P. Pagilla, "Path-constrained optimal trajectory planning for robot manipulators with obstacle avoidance," in *IEEE Int. Conf. Intell. Robots Syst.* Prague, CZ, 2021, pp. *1421–1426*.

[45] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognit.*, vol. 47, no. 6, pp. *2280–2292*, 2014.

[46] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. *1330–1334*, 2000.

[47] D. C. Montgomery, "Cumulative sum and exponentially weighted moving average control charts," in *Introduction to Statistical Quality Control,* 8th ed. Hoboken, NJ, USA: JWS, 2019, ch. 9, pp. *372–399*.

[48] D. C. Montgomery, "Multivariate process monitoring and control," in *Introduction to Statistical Quality Control,* 8th ed. Hoboken, NJ, USA: JWS, 2019, ch. 11, sec. 3, pp. *462–472*.