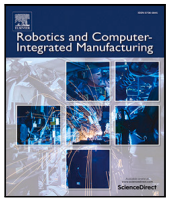
Robotics and Computer–Integrated Manufacturing 74 (2022) 102262

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcm



Full length article

Development of a vision based pose estimation system for robotic machining and improving its accuracy using LSTM neural networks and sparse regression

Diyar Khalis Bilal, Mustafa Unel *, Lutfi Taner Tunc, Bora Gonul

Faculty of Engineering and Natural Sciences, Sabanci University, Tuzla, 34956 Istanbul, Turkey
Integrated Manufacturing Technologies Research and Application Center, Sabanci University, Tuzla, 34906 Istanbul, Turkey

ARTICLE INFO

Keywords:
Machine vision
Pose estimation
Robotic machining
Machine learning
LSTM
Sparse regression

ABSTRACT

In this work, an eye to hand camera based pose estimation system is developed for robotic machining and the accuracy of the estimated pose is improved using two different approaches, namely Long Short Term Memory (LSTM) neural networks and sparse regression. To improve the accuracy obtained from the Levenberg–Marquardt (LM) based pose estimation algorithm, two distinct supervised data driven approaches are proposed which can take the dynamics into account during robotic machining through utilization of the torque information available from the sensors at each joint. The first one is a LSTM neural network and the second one is a method based on sparse regression. The proposed methods are validated by an experimental study performed using a KUKA KR240 R2900 ultra robot while machining a NAS 979 part, during which the orientation of the cutting tool was fixed, and free form milling, during which the orientation of the cutting tool continuously changed. A target object to be tracked by the camera was designed with fiducial markers to guarantee trackability with ±90° in all directions. The design of these fiducial markers guarantee the detection of at least two distinct non-parallel markers from any view, thus preventing pose estimation ambiguities. Moreover, in order to reduce the errors due to the construction of the camera target and placement of the markers on it, this work proposes a method for optimizing the positions of the corners of the fiducial markers in the object frame using a laser tracker. The proposed methods were compared with an Extended Kalman Filter (EKF) and the experimental results show that both of the proposed approaches significantly improve the pose estimation accuracy and precision of the vision based system during robotic machining while proving much more effective than the EKF approach. The attainable absolute position errors were 5.47 mm, 2.9 mm and 2.05 mm on average for NAS 979 machining and 5.35 mm, 2.17 mm and 0.86 mm on average for free form machining when using the EKF, the proposed LSTM network and the proposed sparse regression approaches, respectively. Moreover, the proposed sparse regression based method provides parsimonious models and better results when compared with the proposed LSTM based approach.

1. Introduction

Machining processes are expected to be mostly performed by industrial robots in the near future due to their larger working space, lower prices and flexibility in automation when compared with CNC machines. However, due to their relatively lower accuracy and stiffness they are not being used in high precision fields such as manufacturing. Therefore, development of augmentation systems for increasing the effectiveness of industrial robots in manufacturing parts by increasing the robot's machining accuracy is of utmost importance.

There exists multiple works in literature for increasing the accuracy of industrial robots through static calibration or installation of extra high accuracy encoders at each joint of industrial robots [1,2]. However, static calibration methods do not take disturbances due to machining processes into account and installation of extra encoders is not always feasible, in addition to them being very expensive. A feasible alternative to these is realtime tool path tracking and correction based on visual servoing. In visual servoing various control strategies and visual feedback is used for correcting the pose of a robot's end effector in realtime [3,4]. In the work by Shi et al. [5], a laser tracker based online real-time path compensation method is proposed and validated on a KUKA KR5 arc robot during straight line and circular path tracking. From their experimental results it is shown that absolute

* Corresponding author at: Faculty of Engineering and Natural Sciences, Sabanci University, Tuzla, 34956 Istanbul, Turkey.
E-mail address: munel@sabanciuniv.edu (M. Unel).

https://doi.org/10.1016/j.rcim.2021.102262
Received 20 July 2020; Received in revised form 25 August 2021; Accepted 26 September 2021
Available online 13 October 2021

path tracking error were reduced from 0.744 to 0.014 mm during straight line tracking and from 0.514 to 0.0542 mm during circular path tracking. Qu et al. [6] proposed the usage of a laser tracker in a closed loop feedback system to reduce the pose errors to less than 0.2 mm and 1° in aircraft assembly drilling process. In the work by Shu et al. [7] a position based visual servoing (PBVS) scheme is proposed for dynamic path correction in realtime using C-track 780 photogrammetry sensor. Moreover, the estimated pose by the C-track 780 was smoothed and the effect of image noise and vibrations were reduced through an adaptive Kalman filter. Their proposed method was validated by an experimental study performed during line and circle path tracking in space and they were able to achieve a tracking accuracy of ±0.20 mm for position and ±0.1° for orientation. Moreover, in the work presented in the Comet project [8], the authors were able to reach accuracies of 0.2 mm in robotic milling using an online path compensation based on a Nikon K-CMM photogrammetric sensor and a dynamic compensation mechanism. Additionally, in the work by Moller et al. [9] the absolute accuracy of an industrial milling robot was enhanced using an AICON MoveInspect HR stereo camera system. This camera system has a high accuracy of 50 μm per m^3 measuring volume but has a slow measurement rate of upto 10 Hz as stated in Moller et al.'s work. In their work they used an iterative static control for the pose of the end effector at 10 Hz using a "look then action" type control approach. They demonstrated that using the AICON stereo system one could obtain positioning errors of less than 0.3 mm without performing machining operations.

All these works in literature rely on the availability of a highly accurate external measurement system such as a laser or dynamic photogrammetry tracker, however these sensors are quite expensive. Thus, many researchers proposed utilization of single camera based systems as a more affordable alternative to these sensors. Keshmiri et al. [10] proposed an image based visual servoing (IBVS) trajectory planning method for industrial robots where a monocular camera in an eye in hand configuration is used. Their work is validated on a VS-6556G Denso robot during robot positioning tasks, however they only report errors in image plane rather than Cartesian frame and their algorithm does not operate in realtime. Planar AprilTag markers were attached to the end effector of an industrial robot in the work by Nissler et al. [11] for determining its pose. They reported position tracking errors of less than 10 mm through utilization of optimization techniques. However, their work was not evaluated during trajectory tracking and they did not consider the rank deficiency problem due to the used planar markers. In the work by Liu et al. [12] multi sensor optimal information algorithms (MOIFA) and Kalman filter (KF) were proposed for fusing position information acquired from a photogrammetry system and orientation information obtained from a digital inclinometer. They conducted seventy six positioning experiments in a one meter cube space by using a KP 5 Arc KUKA robot's end effector, without evaluating their approach during trajectory tracking. In the work by Claes et al. [13] structured light was projected on a workpiece and a single camera was attached to the end effector of an industrial robot to estimate the relative pose between the robot and the workpiece. They conducted experiments for robot positioning without performing trajectory tracking and reported a positioning accuracy of 3 mm.

The works in literature utilizing eye in hand approaches for pose estimation are based on the assumption that the industrial robots kinematics or dynamics models are fully known. However, obtaining such models is not a trivial task. In the case of approaches based on KF for vision based pose estimation, a linear dynamic process model is utilized where the measurement and process noise are assumed to be known. However, it is well known in literature that vision based pose estimation and manipulation tasks of industrial robots are nonlinear processes. To overcome these drawbacks in estimation of an industrial robot's end effector extended Kalman filter (EKF) [14,15] and adaptive Kalman filter (AKF) [16,17] have been utilized in literature. However, obtaining accurate dynamic process models for EKF is hard. As for the approaches based on AKF, their effectiveness is limited since these works do not consider the time varying measurement and process noise due to the numerous trajectories followed by the industrial robots at various speeds. This is especially the case in robotic machining where various configurations and trajectories must be utilized for each part to be machined properly. This in turn adds more complexity and uncertainties to the system in addition to the inevitable sensor noise. Therefore, data driven approaches that can take the sensor noise and uncertainties in the system into account have found to be more effective. One of the most effective data based modeling techniques has been proven to be deep learning [18–21]. In this regard, many recent applications of deep learning have emerged in industrial robot applications for system identification of the nonlinear residual errors using a laser tracker [22], in process tool condition forecasting based on a deep learning method [23], prediction of arm trajectories for force limiting on industrial robots for human–robot collaboration [24], and positioning error compensation on two-dimensional manifold for robotic machining [25]. Moreover, data driven approaches have also been used in estimating the pose of human faces from 2D images using Convolutional Neural Networks (CNN) along with multimodal mapping [26]. This pose information can be used for estimating gazing direction which gives important communicative information and visual saliency. In another work deep autoencoders were used to recover the human pose from videos. They construct hypergraph Laplacian with low rank representations to exract unique feature descriptors. Then they use deep autoencoders to nonlinearly map 2D images to 3D poses of humans [27,28].

This work proposes the development of an eye to hand monocular vision based pose estimation system for robotic machining that can be tracked with ±90° from all directions. A camera target (CT) is designed with fiducial markers to prevent ambiguities in pose estimation by guaranteeing detectability of at least two distinct non-parallel markers from any view in a single frame. Moreover, an optimization method is proposed for reducing the inevitable errors in the determination of marker coordinates in object frame due to the construction of the CT and placement of markers on it. Due to the nonlinear dynamic nature of the pose estimation problem and availability of large quantities of data, in this work two distinct supervised data driven modeling approaches are proposed. These approaches can take the dynamics due to robotic machining into account through utilization of the joints torques to improve the pose estimation accuracy based on the Levenberg–Marquardt (LM) algorithm [29]. The first one is a Long Short Term Memory (LSTM) type recurrent neural network (RNN) [30] and the second one is a method based on sparse regression, where in both methods the pose data obtained from a laser tracker is used as the ground truth. Both methods utilize the pose estimated from LM and the joints torques as input and provide an output in terms of pose with a better accuracy and precision. Using either of the proposed methods, one can train the camera based systems using a single laser tracker in a factory where several industrial robots are required to perform the same task.

The rest of this paper is structured as follows: In Section 2, the mathematical foundation of vision based pose estimation, transformations between the coordinate systems and the proposed methodologies are presented. In Section 3 the experimental setup and the construction of the camera target (CT) is described. The effectiveness of the proposed approaches are validated through robotic machining in Section 4, followed by the conclusion in Section 5.

2. Methodology

2.1. Pose estimation using machine vision

Position and orientation (or pose) refer to the 3D translation and rotation which brings the object model wherever it was observed by the camera to the world frame. Therefore, in order to estimate the pose of a camera with respect to a world frame, coordinates of

points both in image plane and the world frame must be known. The camera pose estimation from n 3D-to-2D points correspondences is a fundamental problem in geometric computer vision area. The main goal is to estimate the six degrees of freedom of the camera pose provided that the camera calibration parameters, namely the focal length, the principal point, the aspect ratio and the skew angle are known. In order to compute the pose of a camera one needs a set of 3D-to-2D correspondences between n reference points $P_1 \dots P_n$ expressed in a world coordinate frame W and their corresponding 2D projections $\tilde{p}_1 \dots \tilde{p}_n$ in the image plane as seen in Fig. 1. Therefore, given a 3D point P_i expressed in a world reference frame, and its 2D projection $\tilde{p}_i$ onto the image, we seek to retrieve the pose of the camera with respect to the world frame. Thus, the defined projection can be expressed as:

$$\lambda \tilde{p}_i \;=\; M \; P_i \tag{1}$$

where λ is a nonzero scalar and M is the projection matrix defined as:

$$M = K\,[R\;T] = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} \tag{2}$$

where K is a 3×4 homogeneous matrix which contains the camera intrinsic parameters and $[R\;T]$ is a 4×4 homogeneous matrix which encodes the rigid transformation from the world coordinate system to the camera coordinate system. m_1, m_2 and m_3 are the three rows of the projection matrix. More precisely, K and $[R\;T]$ are defined as:

$$K = \begin{bmatrix} \alpha & s & u_o & 0 \\ 0 & \beta & v_o & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \text{ and } [R\;T] = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tag{3}$$

where the intrinsic parameters α and β are the effective focal lengths in horizontal and vertical directions, s is the skew parameter and u_0 and v_0 are the coordinates of the principal point. R is the rotation matrix and T is the translation vector representing the position and orientation (pose), respectively of the camera coordinate system in world coordinate system. Both the world and the projected image points are represented in homogeneous coordinates as:

$$P_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \text{ and } \tilde{p}_i = \begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{m_1\;P_i}{m_3\;P_i} \\ \frac{m_2\;P_i}{m_3\;P_i} \\ 1 \end{bmatrix} \tag{4}$$

Most of the pose estimation algorithms assume that the intrinsic parameters K of the camera to be known. There exist many works in literature for determination of the intrinsic parameters such as direct linear transformation (DLT) [31] and Zhang's algorithm [32]. However, literature has shown that Zhang's algorithm provides the most accurate results when compared with all of the other camera calibration algorithms. Therefore, in this work the camera system was calibrated with Zhang's algorithm and thus its intrinsic parameters were determined.

Therefore, given a 2D image of an object, and a calibrated camera it is possible to find the pose of the object in camera frame, or alternatively pose of the camera in object frame. Without loss of generality the object frame can be treated as the world frame in the presented formulations. A common way of determining the camera pose is through iterative methods by means of optimization and in this work the method based on Levenberg–Marquardt (LM) [29] was utilized due to its robustness and realtime performance. Such an optimization tries to find a pose that minimizes reprojection error between the detected image points (p_i) and the projected object points ($\tilde{p}_i$) in the world frame. This reprojection error consists of the residual between the detected and reprojected images points in x (rx) and y (ry) coordinates as follows:

$$rx_i = u_i - \tilde{u}_i \tag{5}$$

$$ry_i = v_i - \tilde{v}_i \tag{6}$$

where u_i and v_i are the x and y coordinates of the detected image points, $\tilde{u}_i$ and $\tilde{v}_i$ are the coordinates of the projected image points obtained using Eq. (4). For a set of n correspondences, the reprojection error can be expressed as a vector valued function $f(\boldsymbol{\Phi})$ with $\boldsymbol{\Phi} = (T_X, T_Y, T_Z, \alpha, \beta, \gamma)$, where the first three parameters denote translation and the last three represent orientation. Therefore $f(\boldsymbol{\Phi})$ and the cost function ($F(\boldsymbol{\Phi})$) to be minimized can be defined as follows:

$$f(\boldsymbol{\Phi}) = [rx_1, ry_1, \dots, rx_i, ry_i, \dots, rx_n, ry_n]^T \tag{7}$$

$$F(\boldsymbol{\Phi}) = \frac{1}{2} f(\boldsymbol{\Phi})^T f(\boldsymbol{\Phi}) \tag{8}$$

Afterwards, the pose parameters can be updated iteratively based on the LM algorithm as follows:

$$\hat{\boldsymbol{\Phi}}_k = \hat{\boldsymbol{\Phi}}_{k-1} - (J^T J + \zeta I)^{-1} J^T f(\boldsymbol{\Phi}) \tag{9}$$

where, $\hat{\boldsymbol{\Phi}}_k$ is the estimated pose at current iteration, $\hat{\boldsymbol{\Phi}}_{k-1}$ is the estimated pose at the previous iteration, ζ is a positive damping parameter, I is the identity matrix and J is the Jacobian matrix defined as follows:

$$J \triangleq \frac{\partial f(\boldsymbol{\Phi})}{\partial \boldsymbol{\Phi}} = \begin{bmatrix} \frac{\partial rx_1}{\partial T_X} & \frac{\partial rx_1}{\partial T_Y} & \frac{\partial rx_1}{\partial T_Z} & \frac{\partial rx_1}{\partial \alpha} & \frac{\partial rx_1}{\partial \beta} & \frac{\partial rx_1}{\partial \gamma} \\ \frac{\partial ry_1}{\partial T_X} & \frac{\partial ry_1}{\partial T_Y} & \frac{\partial ry_1}{\partial T_Z} & \frac{\partial ry_1}{\partial \alpha} & \frac{\partial ry_1}{\partial \beta} & \frac{\partial ry_1}{\partial \gamma} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial rx_n}{\partial T_X} & \frac{\partial rx_n}{\partial T_Y} & \frac{\partial rx_n}{\partial T_Z} & \frac{\partial rx_n}{\partial \alpha} & \frac{\partial rx_n}{\partial \beta} & \frac{\partial rx_n}{\partial \gamma} \\ \frac{\partial ry_n}{\partial T_X} & \frac{\partial ry_n}{\partial T_Y} & \frac{\partial ry_n}{\partial T_Z} & \frac{\partial ry_n}{\partial \alpha} & \frac{\partial ry_n}{\partial \beta} & \frac{\partial ry_n}{\partial \gamma} \end{bmatrix} \tag{10}$$

The ζ is controlled at every iteration and through this parameter the LM can switch between gradient descent and Gauss–Newton so as to provide rapid convergence. In summary, the cost function of pose estimation ($F(\boldsymbol{\Phi})$) is minimized iteratively by updating the pose ($\boldsymbol{\Phi}$) until the reprojection error is less than a threshold value. This way the pose of the target object is estimated iteratively each time an image is acquired and the correspondences between the 3D and 2D points are established.

2.2. Transformations between the robot, laser and camera frames

To evaluate the accuracy and precision of the vision based pose estimation, the coordinates of the camera system and a laser tracker were transformed into a common fixed base (FB) frame. The (FB) frame was defined by moving the cutting tool of the robot to a point in world frame and its coordinate was defined as the origin of the (FB) frame. Then a common target's pose, for both laser tracker and the camera system, such as the cutting tool (S) was calculated in the FB frame as follows:

$$\{{}^{FB}T_S\}^C = {}^{FB}T_C \, {}^{C}T_{CT} \, {}^{CT}T_S \tag{11}$$

$$\{{}^{FB}T_S\}^L = {}^{FB}T_L \, {}^{L}T_{LT} \, {}^{LT}T_S \tag{12}$$

where ${}^{CT}T_S$ is the fixed rigid body transformation between the cutting tool and the camera target frames, ${}^{LT}T_S$ is the fixed rigid body transformation between the cutting tool and the laser target frames, ${}^{L}T_{LT}$ and ${}^{C}T_{CT}$ are provided by the measurements from the laser tracker and the camera based pose estimation, respectively. Therefore, the pose of the cutting tool estimated by the camera system and measured by the laser tracker can be compared with each other in the FB frame. The transformations between the coordinates of the targets and trackers are given in Fig. 2.

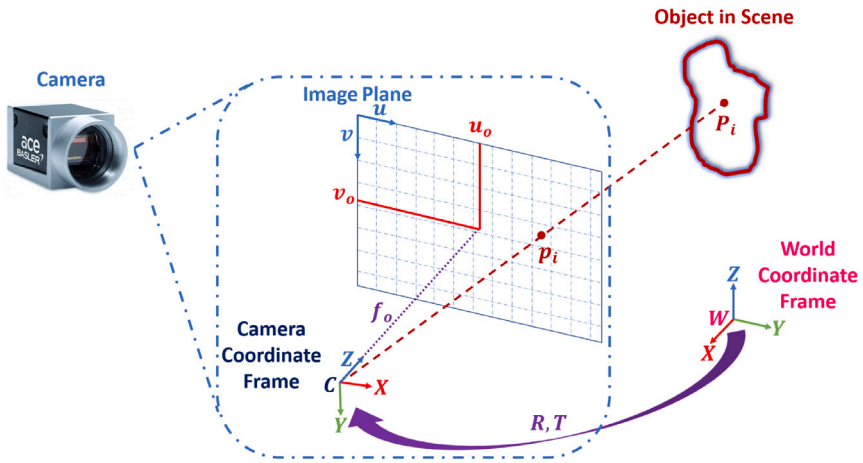


Fig. 1. Projection of 3D point onto 2D image plane using the pinhole camera model.

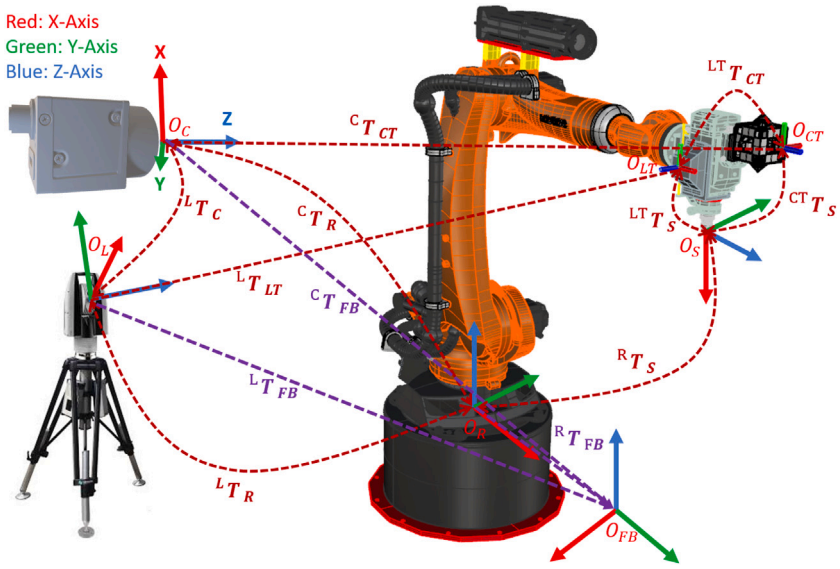


Fig. 2. Coordinate system transformations, where ${}^{A}T_{B}$ is a transformation representing B in A and O_A denotes origin of frame A. C, L, R, CT, LT, S and FB denote Camera, Laser, Robot, Camera Target, Laser Target, Cutting Tool and Fixed Base frames.

2.3. Improved vision based pose estimation using LSTM neural network

In this work, a supervised machine learning approach for increasing the accuracy and precision of camera based pose estimation is proposed by training it with the pose measurements $(T_X, T_Y, T_Z, \alpha, \beta, \gamma)$ obtained from a laser tracker. Formulation of this problem as a machine learning framework requires careful selection of inputs and ground truth data. Th ground truth pose data can be obtained from the laser tracker and the inputs can be provided by the vision based system in the form of the estimated pose $(\hat{T}_X, \hat{T}_Y, \hat{T}_Z, \hat{\alpha}, \hat{\beta}, \hat{\gamma})$ as discussed in Section 2.1. However, just using these as inputs may not be enough for machining processes, since it is known that the pose of the cutting tool of the robot can deviate from its path due to the forces and vibrations induced during the machining process. These forces and vibrations will be manifested in the exerted torques by the robot that can be measured by the torque sensors at the robot's joints. Moreover, the dynamics of the robot itself manifests itself in the joint torques. Thus, the torques $(\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6)$ can be utilized as additional inputs so as to take the induced forces and the dynamics due to machining into account during the pose estimation process.

Several machine learning architectures exist in literature for modeling dynamic systems and one such approach is LSTM type recurrent neural network (RNN). LSTM was originally developed for word or sentence prediction and it proved to provide robust models for such problems. However, this work proposes to use it with time varying data such as the estimated pose by the vision based system. It is well known that the problem of pose estimation is a continuous and time dependent regression problem. Therefore, in this work a LSTM is proposed to extract the dynamics from the input data and use it as input to a regression type neural network to provide a more accurate and precise pose. The proposed machine learning architecture for improving vision based pose estimation is shown in Fig. 3 where twelve inputs are fed into the LSTM network at each time frame. Six of the inputs are the pose data estimated by the LM based algorithm and the other six are the torques obtained from the torque sensors at each joint of the robot. The input to the proposed machine learning framework is denoted by X_t and contains twelve elements at each time frame. This input is then concatenated with the state vector of the previous time frame S_{t-1} with a chosen size and it goes into three gates known as input (R_t), forget (F_t) and output (O_t) gates defined as follows:

$$R_t = \sigma(W_R[S_{t-1}, X_t]) \tag{13}$$

$$F_t = \sigma(W_F[S_{t-1}, X_t]) \tag{14}$$

$$O_t = \sigma(W_O[S_{t-1}, X_t]) \tag{15}$$

where σ is the sigmoid activation function and W_R, W_F & W_O are the weight matrices associated with the input, forget, and output gates respectively.

Through the usage of these gates the LSTM network can learn which information to keep or discard during the training phase of the network. This way the short-term memory problems seen in vanilla RNN networks can be prevented. The memory of the network is stored by the cell/memory state (C_t) which enables the information learned during the earlier time steps to make it through to the latest time steps and is defined as follows:

$$C_t = F_t \odot C_{t-1} + R_t \odot \widetilde{C}_t \tag{16}$$

where C_{t-1} is the previous cell/memory state, $\odot$ denotes element wise product and $\widetilde{C}_t$ is another gate with the associated weight matrix W_C and is defined as:

$$\widetilde{C}_t = \tanh(W_C[S_{t-1}, X_t]) \tag{17}$$

where tanh(.) is the tangent hyperbolic activation function and W_C is the associated weight matrix.

After the relevant information is extracted from the input features through utilization of the aforementioned gates, the cell state (C_t) and the output gate (O_t) can the be used to calculate the state vector (S_t) as follows:

$$S_t = O_t \odot \tanh\left(C_t\right) \tag{18}$$

The complexity of the problem determines the number of states to be used in the state vector and as such it is user defined. The state vector is then fed into a single layer regression network with linear activation function to estimate the corrected pose ($\widehat{\Omega}_t$) of the target as follows:

$$\widehat{\Omega}_t = W S_t + B \tag{19}$$

where, W and B are the weight matrix and the bias vector associated with the regression layer.

Afterwards, the cost function is defined as the Euclidean norm of the error between the ground truth pose provided by the laser tracker (Ω_t^i) and the estimated corrected pose ($\widehat{\Omega}_t^i$) obtained from the output of the regression layer as follows:

$$CF_L = \frac{1}{N}\sum_{i}^{N} \|\Omega_t^i - \widehat{\Omega}_t^i\|_2 \tag{20}$$

The proposed method was trained using Adam optimizer and it was coded in TensorFlow [33] software. The number of states and neurons used in the LSTM and regression networks are given in the experimental results section for different machining processes.

2.4. Improved vision based pose estimation using sparse regression

In this section a simpler and more intuitive approach based on sparse regression is presented for improving the accuracy and precision of vision based pose estimation. This method also utilizes the laser tracker as ground truth and the pose estimated by the LM based algorithm and torque information as input. This work builds upon the work presented for discovering governing dynamical equations from data through sparse identification of nonlinear dynamics (SINDy) by Brunton et al. [34]. Their work leverages the fact that the dynamics of a physical system can be usually defined through a few terms. Therefore, making the defining equations sparse in a high dimensional nonlinear function space. In their work a large amount of data is collected to define the equations of motion for dynamic systems through state space functions. They generate some candidate nonlinear functions such as constants, higher order polynomials, sinusoidal functions, ..., etc. by collecting a time history of the state $X(t)$ and its derivative. They formulate the problem as sparse regression and propose a method based on sequential thresholded least-squares algorithm [34] to solve it. Their proposed approach is a faster and robust alternative to the least absolute shrinkage and selection operator (LASSO) [35] which is an $\ell 1$-regularized regression which promotes sparsity. Their proposed approach can be used to determine the sparse vectors of coefficients that define the dynamics of a physical system, thus it allows to obtain parsimonious models that balance accuracy with model complexity to avoid overfitting.

However, in this work the sparse regression problem is formulated for Nonlinear Finite Impulse Response (NFIR) model, hence the proposed method is termed as SNFIR. In particular, the relationship between the groundtruth pose provided by the laser tracker and the pose estimated by the vision system in addition to the induced torques on the robot's joints while machining is assumed to be represented by the following NFIR dynamic model with upto k regressors (see Eqs. (21)–(26) in Box I) where $x_1(t)$ to $x_6(t)$ are the $\hat{T}_X, \hat{T}_Y, \hat{T}_Z, \hat{\alpha}, \hat{\beta}$, and $\hat{\gamma}$ estimated by the LM based pose estimation algorithm at the current time step, $x_7(t)$ to $x_{12(t)}$ are the torques of the robot joints measured by the robot's torque sensors at the current time step, $X(t-k)$ are the estimated LM based pose and the joint torques at the kth preceding time step, $y_1(t)$ to $y_6(t)$ are the ground truth $T_X, T_Y, T_Z, \alpha, \beta$, and γ measured by the laser tracker at the current time step, Δ contains the sparse vectors of coefficients, X^{P_2} denotes the quadratic nonlinearities in the inputs $(X(t), X(t-1), \ldots, X(t-k))$, and $\Gamma(X(t), X(t-1), \ldots, X(t-k))$ is the library consisting of candidate nonlinear functions.

A candidate function that defines the relationship between the ground truth pose and the input is represented by each column of the augmented library $\Gamma(X(t), X(t-1), \ldots, X(t-k))$. These functions can be freely chosen and in this work a constant and up to 2nd order polynomials (X^{P_2}) were chosen with cross terms. As for the number of input regressors, in this work setting k to 1 proved sufficient for obtaining the best results while having least impact on the processing time. Therefore, the resulting size of the sparse regression problem using m samples is as follows:

$$Y(t)_{mx6} = \Gamma(X(t)_{mx12}, X(t-1)_{mx12})_{mx325}\Delta_{325\times 6} \tag{27}$$

The sequential thresholded least-squares based algorithm proposed by Brunton et al. [34] starts with finding a least squares solution for Δ and then setting all of its coefficients smaller than a threshold value (λ) to zero. After determining the indices of the remaining nonzero coefficients, another least squares solution for Δ onto the remaining indices is obtained. This procedure is performed repeatedly for the new coefficients using the same λ until the nonzero coefficients converge. This algorithm is computationally efficient and rapidly converges to a sparse solution in a small number of iterations. Moreover, only a single parameter λ is required to determine the degree of sparsity in Δ. The overall flowchart of the proposed method is shown in Fig. 4.

2.5. Optimization of object points in camera frame using a laser tracker

In order to estimate the pose of a target object with respect to the camera frame, a camera target (CT) must be designed and fitted with markers with predefined and known locations in object frame. These locations can be obtained from its CAD model. The locations of the constructed CT's points such as corners of some markers in the object frame needs to be known precisely so as to obtain an accurate and precise pose. However, due to the inevitable errors in the construction of the CT and placement of the markers on it, the locations of the markers will differ from the ones obtained from the CAD model. Therefore, this work proposes to determine the locations of the object points precisely by utilizing a laser target through optimization. The proposed optimization relies on the usage of the model given in (1) which relates the object points P in object frame to their projections $\tilde{p}$ on the image plane by utilizing the pose of the camera target in camera frame as provided by the laser tracker $\{{}^{C}T_{CT}\}^{L}$. Therefore, in

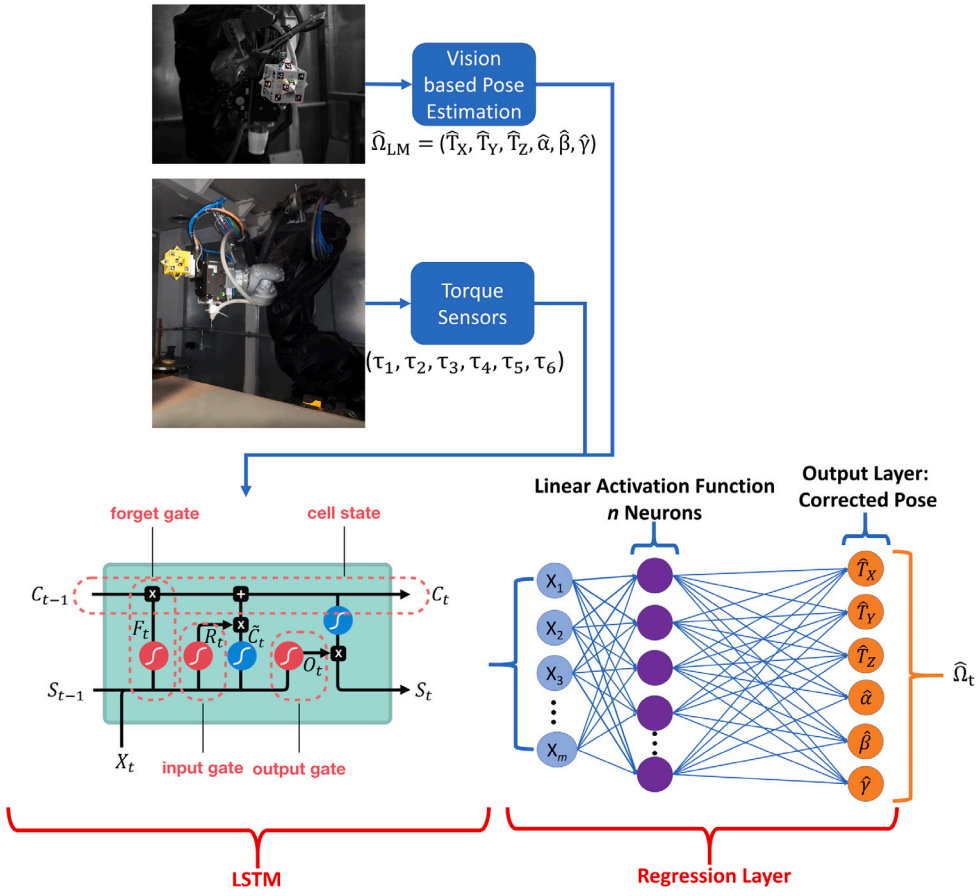


Fig. 3. The proposed neural network architecture for improving vision based pose estimation.

$$Y(t) = \Gamma(X(t), X(t-1), \ldots, X(t-k))\Delta \tag{21}$$

where

$$X(t) = \begin{bmatrix} x_1(t_1) & \cdots & x_{12}(t_1) \\ \vdots & \ddots & \vdots \\ x_1(t_m) & \ldots & x_{12}(t_m) \end{bmatrix} \tag{22}$$

$$X(t-k) = \begin{bmatrix} x_1(t_1-k) & \cdots & x_{12}(t_1-k) \\ \vdots & \ddots & \vdots \\ x_1(t_m-k) & \ldots & x_{12}(t_m-k) \end{bmatrix} \tag{23}$$

$$Y(t) = \begin{bmatrix} y_1(t_1) & \cdots & y_6(t_1) \\ \vdots & \ddots & \vdots \\ y_1(t_m) & \ldots & y_6(t_m) \end{bmatrix} \tag{24}$$

$$\Gamma(X(t), X(t-1), \ldots, X(t-k)) = \begin{bmatrix} 1 & X(t) & X(t-1) & \cdots & X(t-k) & X(t)^{P_2} & X(t-1)^{P_2} & \cdots & X(t-k)^{P_2} \end{bmatrix} \tag{25}$$

$$X(t)^{P_2} = \begin{bmatrix} x_1^2(t_1) & x_1(t_1)x_2(t_1) & \cdots & x_2^2(t_1) & x_2(t_1)x_3(t_1) & \ldots & x_{12}^2(t_1) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \\ x_1^2(t_m) & x_1(t_m)x_2(t_m) & \ldots & x_2^2(t_m) & x_2(t_m)x_3(t_m) & \ldots & x_{12}^2(t_m) \end{bmatrix} \tag{26}$$

Box I.

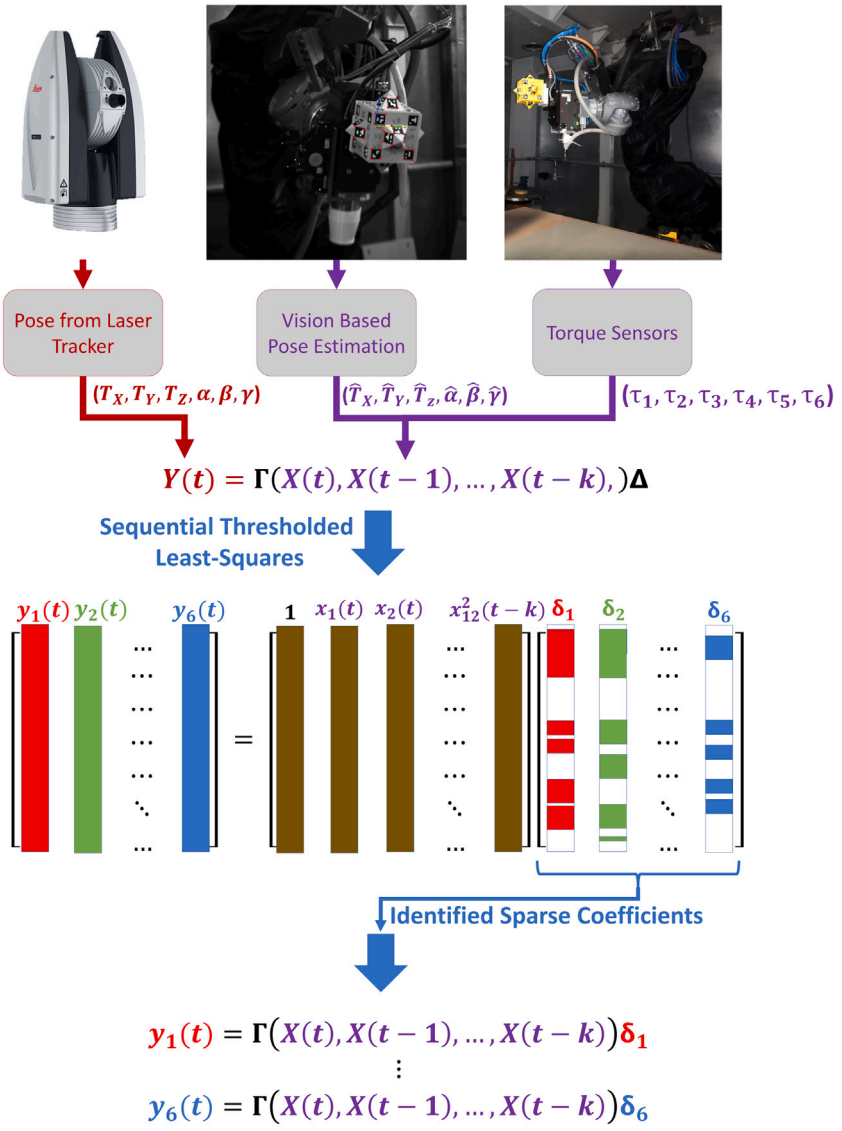


Fig. 4. The proposed Sparse Nonlinear Finite Impulse Response (SNFIR) model for improving vision based pose estimation.

this work the $\bar{P}$ matrix containing Q corners points (object points) of the M markers were optimized using the Adam [36] backpropagation algorithm, where the initial solution to the optimizer was provided from the CAD model and the corresponding image points of the corners (p) were obtained from the marker detection algorithm. The $\bar{P}$ matrix contains the $X, Y,$ and Z coordinates of each corner in object frame and is concatenated with 1 in order to homogenize it. Moreover, since each corner point needs to be optimized individually, their unique identities (D) were used to optimize the object points. The D matrix is $N \times Q$ containing the one-hot encoded form of the unique identities of Q corners for N samples. Each time the optimization is run, it is run for a single acquired sample (image), thus D_i becomes a vector with a size of $1 \times Q$. To clarify it more, when an image is acquired and it contains only corners with identities 8 to 28, then all of the elements of vector D_i are zeros except at the locations of 8 to 28, which are 1. This way, during optimization only the detected corners are optimized, and the undetected ones are not changed since their respective errors are zero. This way the optimization is performed progressively for $i = 1, 2, \ldots, N$ acquired samples based on the following cost function:

$$CF_O = \| (D_i \odot p) - \hat{p} \|_F \tag{28}$$

where CF_O is the cost function defined to be the Frobenius norm of the matrix with $D_i \in R^{1\times Q}$ containing the one hot encoded form of the identities of the detected corner points in current image, $\odot$ denoting element wise product, $p \in R^{3\times Q}$ containing the ground truth image points of the corners obtained from the marker detection algorithm and $\hat{p} \in R^{3\times Q}$ containing the normalized image points of the corners in image plane defined as:

$$\hat{p} = \begin{bmatrix} u/w \\ v/w \\ w/w \end{bmatrix} \text{ and } \bar{p} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = K\{{}^{C}T_{CT}\}_i^L \left(D_i \odot \bar{P}\right) \tag{29}$$

where $K \in R^{3\times4}$ is the intrinsic matrix obtained via camera calibration, $\bar{P} \in R^{4\times Q}$ contains the corner points to be optimized in camera target's frame and $\{{}^{C}T_{CT}\}_i^L \in R^{4\times4}$ is the ground truth pose of the camera target in camera frame obtained from the laser tracker for the current image as follows:

$$\{{}^{C}T_{CT}\}^L = {}^{C}T_{FB}\,{}^{FB}T_{L}\,{}^{L}T_{LT}\,{}^{LT}T_{CT} \tag{30}$$

3. Experimental setup and design of the camera target for pose estimation

In this work a KR240 R2900 ultra KUKA industrial robot, a Basler acA2040-120um camera and a Leica AT960 laser tracker as shown in Fig. 5 were used in the experimental setup. The KUKA KR240-2900

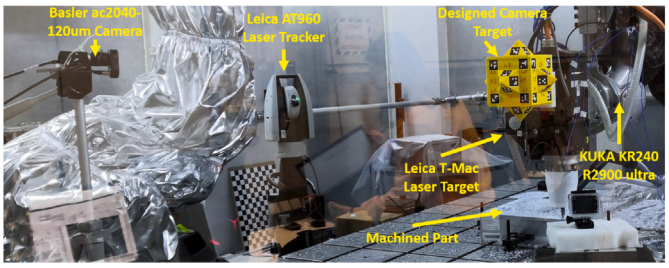


Fig. 5. The experimental setup.

robot was used for machining and the laser tracker was used to track the robot's cutting tool in realtime by using the T-MAC probe rigidly attached to the end effector with an accuracy of ±10 micrometers. To track the cutting tool of the robot and estimate its pose from the camera, a target object was designed with fiducial markers and it was fixed to the end effector of the robot. These markers were chosen to be generated using the ArUco library which generate 2D fiducial markers that can be detected robustly in realtime. The algorithm proposed by Romero-Ramirez et al. [37] can provide the locations of these markers at subpixel accuracy in image plane, which is crucial for vision based pose estimation.

The camera target (CT) was constructed using 3D printing and was designed to have 5 faces and hold 8 markers in each face. Four of the markers in each face were placed in parallel with each other and the remaining four were designed to be at 60° with the first four markers. This was done to avoid the usage of markers from the same plane in pose estimation algorithms. It has been proven in literature that if points from at least two distinct non-parallel planes are used in pose estimation algorithms, then they can prevent pose ambiguities and provide a unique solution. The CT's dimensions are $250 \times 234 \times 250$ mm and it weighs 500 g. The CT was made to be modular and only its base is permanently mounted on the robot. The CT slides into this base during operation. The ArUco markers were generated from ArUco's $4 \times 4 \times 100$ library and fixed to the holes made in each face of the CT as shown in Fig. 6 where each marker's unique ID is given beside them. Through the usage of this CT, the coordinates of all the markers in the object frame can be obtained from the CAD model to be used in pose estimation algorithms.

4. Experimental results

4.1. Detection of the camera target and optimization of object points

The realtime vision based pose estimation and synchronization of data between the laser tracker, the robot and the camera system was realized using LabVIEW [38] software. A Basler ac2040-120um camera was used to acquire images with a resolution of 640 × 480 pixels at 375 Hz. The acquired images were then fed into the python [39] node inside LabVIEW to detect the markers and estimate the pose using the ArUco marker detection and Levenberg–Marquardt algorithms, respectively. The overall communication protocols and pose estimation flow chart of the system is shown in Fig. 7. Each of these algorithms can operate at 1000 Hz and Fig. 8 shows the obtained results for detection of the markers' corners and the estimated pose of the camera target (CT) from multiple views. These results clearly show that with the usage of the designed CT multiple non-planar markers can be detected from a single view with a viewing angle of ±90° from all sides, therefore the ambiguities in pose estimation due to the rank deficiency problem is prevented.

As stated in Section 2.5, the locations of the object points in object frame were initially acquired from the CAD model. However, due to the errors in construction of the camera target and placement of the

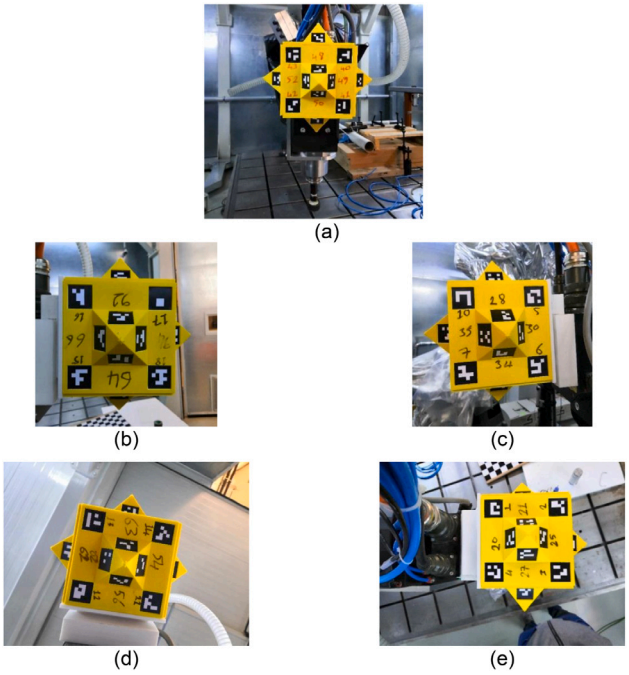


Fig. 6. (a) Front, (b) Left, (c) Right, (d) Bottom and (e) Top views of ArUco markers placed on the target to be tracked.

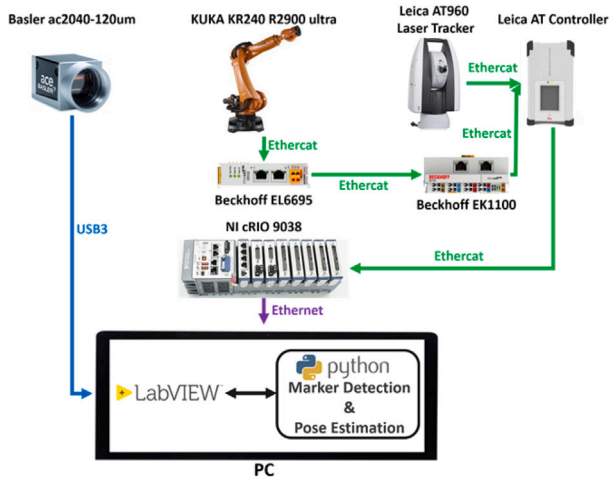


Fig. 7. Experimental setup communication protocols and pose estimation flowchart.

markers, this work proposes to correct the object points through optimization. In order to acquire data for the optimization, the constructed camera target (CT) was rotated ±30 around X, Y and Z axes, as shown in Fig. 9, while data was collected from the laser and camera. The rotation was limited to ±30 due to the laser tracker's limitations and the physical placement of the tracker and camera in the lab. Afterwards, the optimization proposed in Section 2.5 was performed on the acquired data for $M = 40$ markers and $Q = 160$ corner points for which the results shown in Fig. 10 were obtained. From these results it is observed that the average, maximum and minimum of the mean squared error in reprojected image points is lowered by 0.17, 1.185 and 0.07 pixels, respectively when using the optimized object points. These correspond to 10.9%, 5.8% and 92.6% improvements, respectively and they are necessary to obtain the best estimates through the LM algorithm.

Fig. 8. (a)–(f) Samples showing marker detection (detected corners are in red) and estimated pose (red, green, blue coordinate axes) of the target object with respect to the camera frame. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2. Improved vision based pose estimation using LSTM neural networks and sparse regression

After optimizing the object points, the accuracy and precision of the camera system was evaluated during robotic machining of aluminum blocks. The machining processes were chosen to be a NAS 979 part [40], during which the orientation of the cutting tool was fixed, and free form milling, during which the orientation of the cutting tool continuously changed for each axis individually.[1] The details of the NAS 979 machining processes are given in Table 1 and for the free form are given in Table 2. In the experiments the pose estimation was run in realtime but due to the acquisition rate limitations of the torque data acquired from the robot, the frame rate for pose estimation by the camera system and the pose data acquired from the laser tracker were limited to 10 frames per second.

Table 1
Robotic milling processes with no change in cutting tool orientation.

Process name	Milling - 1	Milling - 2	Hole milling
Milling type	Climb cut	Climb cut	Climb cut
Tool type	Flat end mill (2 flutes)	Flat end mill (2 flutes)	Flat end mill (2 flutes)
Diameter of the Tool	16 mm	16 mm	16 mm
Axial and radial depth cuts	1 and 12 mm	1 and 12 mm	0.5 mm (Helical cut)
Spindle speed	5968 rpm	5968 rpm	5968 rpm
Feed rate	1193.6 mmpm	1193.6 mmpm	596.8 mmpm
Acquired data size	19295 samples	25820 samples	5472 samples

Afterwards, both of the proposed methods were used to improve the pose estimated by the LM based pose estimation algorithm with

[1] The dataset of these experiments is available at https://github.com/diyarbilal/Dataset-for-Improving-Vision-Based-Pose-Estimation-During-Robotic-Machining.

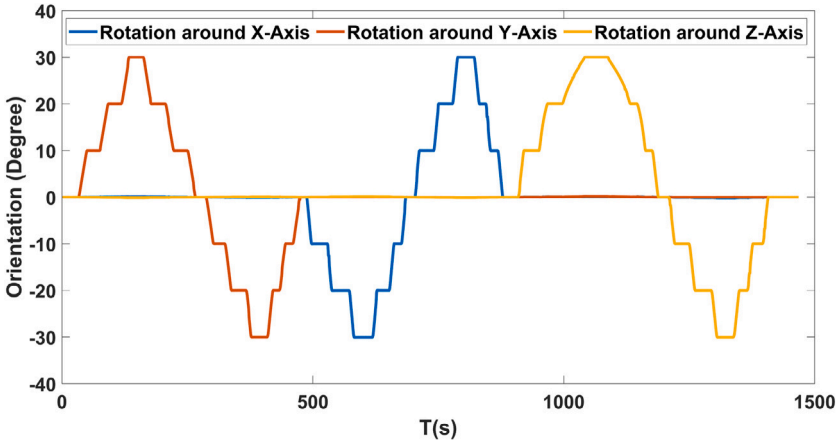


Fig. 9. Trajectory of the camera target for optimization of object points.

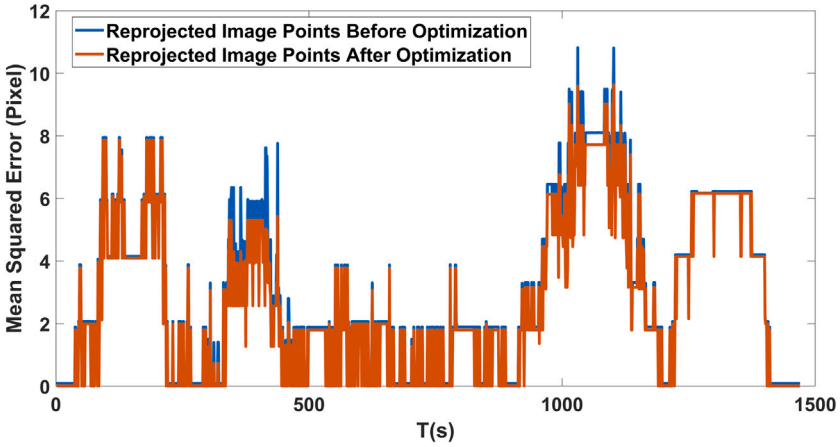


Fig. 10. Mean squared errors in reprojected image points, before vs after optimization of object points.

Table 2
Robotic milling processes with changes in cutting tool orientations.

Process name	Milling - Roll Axis	Milling - Yaw Axis	Milling - Pitch Axis
Milling type	Climb cut	Climb cut	Climb cut
Tool type	Ball mill	Ball mill	Flat end mill (2 flutes)
Diameter of the tool	12 mm	12 mm	16 mm
Axial and radial depth cuts	0.5 and 9 mm	0.5 and 9 mm	0.5 and 12 mm
Spindle speed	6000 rpm	6000 rpm	6000 rpm
Feed rate	1800 mmpm	1800 mmpm	1200 mmpm
Acquired data size	45943 samples	83153 samples	33542 samples

utilization of the torques acquired from the robot's joints as discussed in Sections 2.3 and 2.4. Both of the proposed methods were trained and validated three times for each machining process based on the time series cross validation [41] approach. For this purpose, initially the network was trained using 30% of the data and validated on the rest, then it was trained with 50% and validated on the rest and finally it was trained using 30% of the data and validated on the remaining 70%. Thus, the validation results were calculated by taking the average of the aforementioned three validation cases for each machining process. Moreover, the effectiveness of the proposed approaches are compared with an Extended Kalman Filter (EKF, [42]) which is widely used in literature for nonlinear time series prediction. In this work both the LM based pose and the torque information were utilized as measurement inputs to the EKF. Initially the machining of the NAS 979 part was performed as shown in Fig. 11. In order to determine the best hyperparameters of the proposed LSTM and SNFIR approaches for the NAS 979 part, the proposed methods were trained with 70% of the data and validated on the remaining 30% multiple times with varying state sizes and lambda parameter for the LSTM and SNFIR approaches, respectively. The training of the proposed LSTM network for this machining process was performed using 1000 mini batches for 20000 iterations while varying the state size and fixing the unit size to 6 in the fully connected layer. As for the SNFIR approach the number of iterations was fixed to 10 while varying the lambda parameter. The obtained results for Milling-1 machining process are shown in Tables 3 and 4 where the absolute errors were calculated as follows:

$$E_T = \sqrt{E_X^2 + E_Y^2 + E_Z^2} \tag{31}$$

$$E_R = \sqrt{E_{Roll}^2 + E_{Pitch}^2 + E_{Yaw}^2} \tag{32}$$

where E_X, E_Y, E_Z, E_{Roll}, E_{Pitch}, and E_{Yaw} are the absolute errors between the estimated and ground truth pose provided by the laser tracker, and E_T & E_R are the absolute errors for position and orientation trajectories.

As observed from these results, the usage of more than 10 states does not further significantly improve the pose therefore, 10 states was found to be sufficient. Thus, the training of the proposed LSTM network was chosen to be performed using 1000 mini batches for 20000 iterations using 10 states for the LSTM and 6 nodes at the fully connected layer for the NAS 979 machining process. As for the proposed SNFIR model, as seen from these results, setting the λ parameter to 0.0001 while performing 10 iterations proved to provide the best results. The effectiveness of the proposed approaches is demonstrated by the results

Table 3
Hyperparameter tuning for the LSTM approach for Milling - 1 machining process.

Machining process	LSTM state size	Validation errors (30% of the dataset)			
		Mean		Standard deviation	
		E_T (mm)	E_R (°)	E_T (mm)	E_R (°)
Machining 1	6	10.341	0.047	8.128	0.030
	8	5.186	0.044	4.798	0.029
	10	3.213	0.041	2.876	0.029
	12	3.210	0.041	2.875	0.029

Table 4
Hyperparameter tuning for the SNFIR approach for Milling - 1 machining process.

Machining process	Lambda	Validation errors (30% of the dataset)			
		Mean		Standard deviation	
		E_T (mm)	E_R (°)	E_T (mm)	E_R (°)
Machining 1	0.01	2.364	0.027	2.185	0.022
	0.001	2.107	0.027	2.007	0.022
	0.0001	2.079	0.027	1.976	0.021
	0.00001	2.084	0.027	1.977	0.021

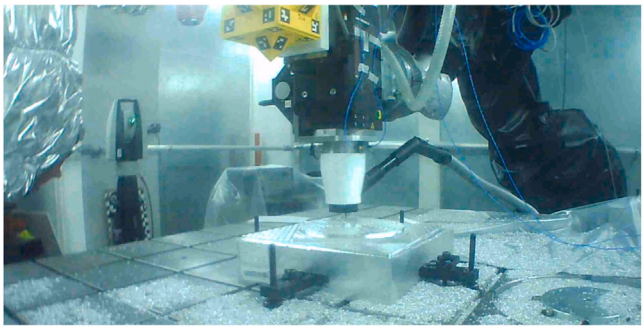


Fig. 11. NAS 979 robotic machining process.

shown in Figs. 12 to 17 and Tables 7 to 9 during machining of the NAS 979 part. These figures are plotted using the pose data acquired from the realtime LM based pose estimation during machining in addition to the improved pose obtained through the proposed methods. Figs. 12, 14 and 16 show the position tracking results of the robot's cutting tool using the EKF, the proposed LSTM neural network and the proposed SNFIR model. In these figures, the trajectory of the cutting tool as tracked by the laser tracker is shown in blue. The yellow trajectories from left to right are the improved poses through utilization of the EKF, the proposed LSTM network and the SNFIR model, respectively. As seen from these figures, when pose from LM and torques are provided as input to both of the proposed methods, the resultant position trajectory is closer to the ground truth provided by the laser tracker when compared with the EKF approach. Moreover, the orientation errors around X, Y and Z axes, hereby denoted as Roll, Pitch and Yaw, respectively, are shown in Figs. 13, 15 and 17 for the EKF, LSTM and SNFIR based methods. The absolute errors for position (E_T) and orientation (E_R) as well as their standard deviation for the NAS 979 part are tabulated in Tables 7 to 9 for the three stages of the NAS 979 machining. These results show that the orientation errors are also reduced when both the pose from LM and the torque information is provided as input to the proposed methods.

As for the effectiveness of the proposed approaches for machining during changing cutting tool orientations, a free form machining was performed in three stages as tabulated in Table 2. During each stage, the orientation of a single axis was changing and the other two was fixed so as to evaluate the accuracy of the camera based pose estimation system in each scenario. Similar to the NAS 979 machining, the proposed methods' hyperparameters for milling with change in orientations were determined by training the proposed LSTM network using 20000 mini batches for 20000 iterations while varying the state size and fixing the unit size to 12 in the fully connected layer. As for the SNFIR approach the number of iterations was fixed to 10 while varying the lambda parameter. The obtained results for Milling-Roll machining process are shown in Tables 5 and 6.

Table 5
Hyperparameter tuning for the LSTM approach for Milling - Roll machining process.

Machining process	LSTM state size	Validation errors (30% of the dataset)			
		Mean		Standard deviation	
		E_T (mm)	E_R (°)	E_T (mm)	E_R (°)
Machining roll	12	1.938	0.642	1.779	0.568
	16	1.438	0.442	1.379	0.324
	20	1.224	0.336	1.004	0.273
	24	1.222	0.334	1.002	0.272

Table 6
Hyperparameter tuning for the SNFIR approach for Milling - Roll machining process.

Machining process	Lambda	Validation errors (30% of the dataset)			
		Mean		Standard deviation	
		E_T (mm)	E_R (°)	E_T (mm)	E_R (°)
Machining roll	0.01	0.686	0.136	0.580	0.113
	0.001	0.621	0.132	0.523	0.109
	0.0001	0.571	0.124	0.480	0.104
	0.00001	0.574	0.124	0.482	0.104

Based on these results, the training of the proposed LSTM network for changing orientations was chosen to be performed using 20000 mini batches for 20000 iterations using 20 states for LSTM and 12 nodes at the fully connected layer. This is because the utilization of larger number of states does not provide meaningful improvement and it will reduce the processing time. As for the proposed SNFIR model, the training was chosen to be performed for 10 iterations using a threshold value (λ) of 0.0001 since this value provides the best results. The machined part in free form is shown in Figs. 18–20. Similar to the NAS 979 machining, the proposed methods for the free form machining were trained with 30%, 50% and 70% of the data and validated on the remaining parts. Then, the validation results were calculated by taking the average of the aforementioned three validation cases for each machining process. The tracked trajectory of the cutting tool of the robot are shown in Figs. 21, 23 and 25 as tracked by the laser tracker and improved through the EKF, the proposed LSTM network and SNFIR model. Moreover, the orientation errors are shown in Figs. 22, 24 and 26. The absolute errors of the estimated pose in the case of free form machining are tabulated in Tables 10 to 12. All of the obtained results in the case of NAS 979 and free form machining are discussed in detail in Section 4.3.

4.3. Discussion of the experimental results

In this section, the results obtained for improving vision based pose estimation through the proposed LSTM and sparse regression approaches are discussed in detail. As mentioned in Section 4.2, the accuracy and precision of the proposed approaches were first evaluated during the machining of a NAS 979 part. During this machining process the orientation of the cutting tool was fixed and did not change. The machining was divided into three parts as given in Table 1 due to the breakage of the cutting tool during the machining process. Figs. 12 and 13 show the results obtained through the proposed LSTM neural network, the proposed SNFIR model and compared with EKF for the first part of the NAS 979 machining. Moreover, Table 7 shows the obtained errors and standard deviation of errors for the Milling - 1 machining. As discussed in Section 4.2, these results are obtained based on time series cross validation during which the proposed methods were trained using 30%, 50%, and 70% of the dataset and were validated on the remaining

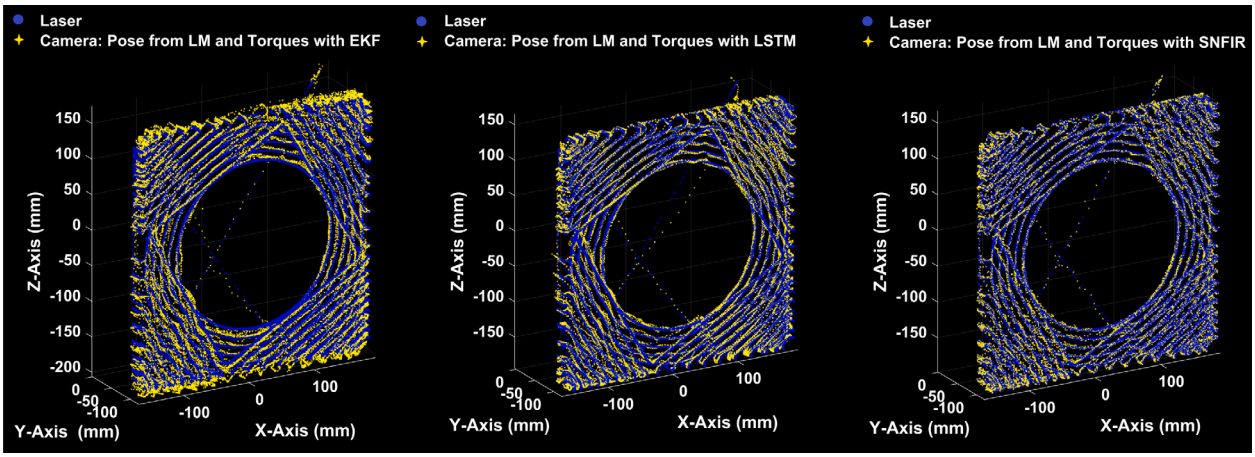


Fig. 12. Results obtained for tracking the position of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model with comparison to laser tracker during machining of Milling - 1. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

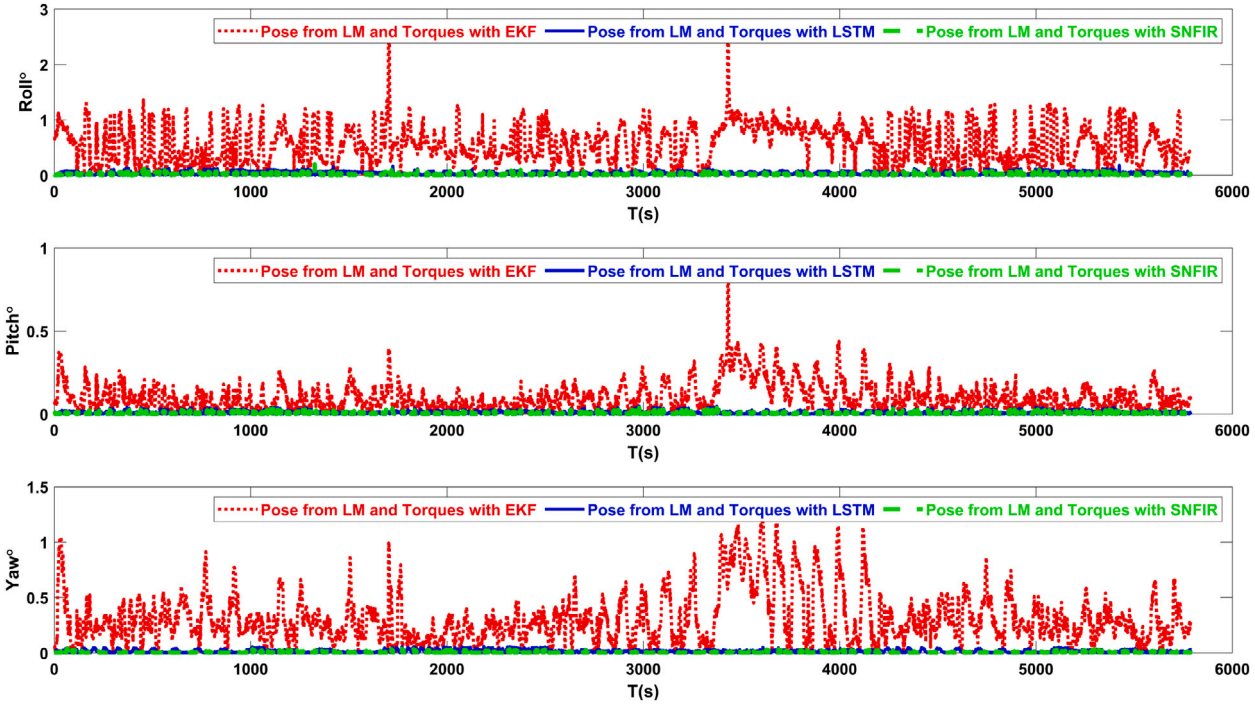


Fig. 13. Errors for tracking the orientation of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model during machining of Milling - 1.

Table 7
Pose tracking errors based on time series cross validation during machining, Milling - 1.

Algorithm	Validation error	Mean	Improvement percentile (%)	Standard deviation	Improvement percentile (%)
LM	ET (mm) ER (°)	6.42 0.67		4.76 0.43	
LM and torques with EKF	ET (mm) ER (°)	5.98 0.66	6.92 1.43	4.59 0.42	3.70 3.68
LM and torques with LSTM	ET (mm) ER (°)	3.00 0.04	53.26 93.71	2.68 0.03	43.85 93.25
LM and torques with SNFIR	ET (mm) ER (°)	2.12 0.03	66.98 95.20	1.98 0.02	58.35 95.01

data. Afterwards, the average of these three validations was taken to calculate the validation errors. As observed, the EKF was able to improve the position tracking (E_T) by only 6.92% and the orientation tracking (E_R) by only 1.43% when compared with the pure LM based approach. Moreover, the standard deviation of errors were reduced by 3.70% and 3.68% for position and orientation tracking, respectively. As for the proposed LSTM network, it was able to significantly improve the position and orientation tracking by 53.26% and 93.71%,

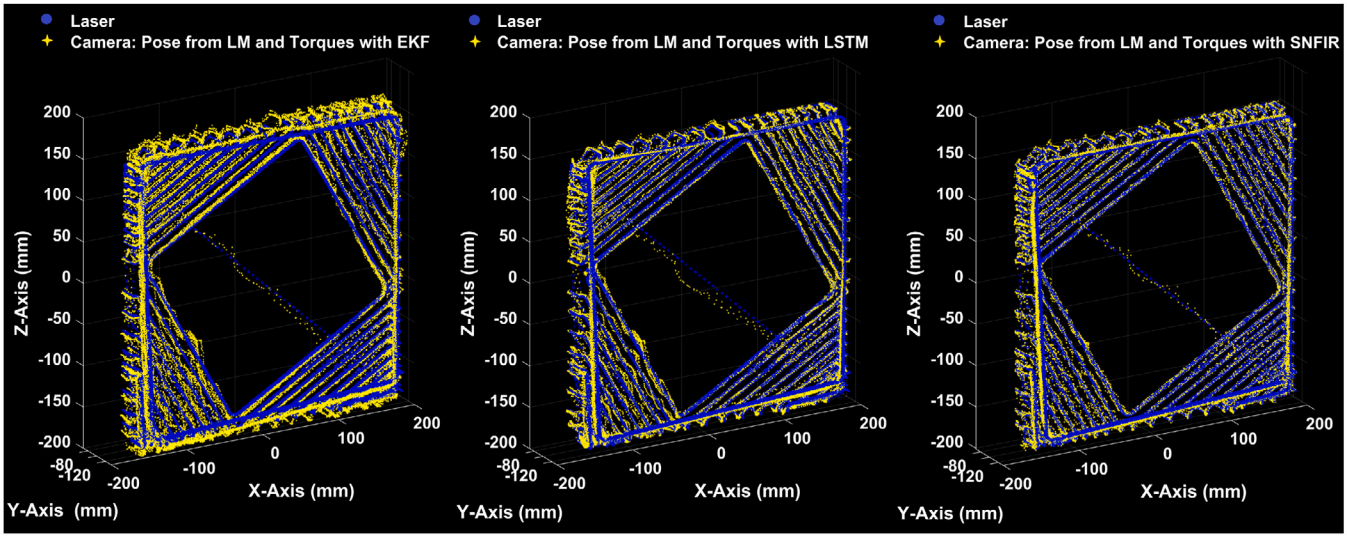


Fig. 14. Results obtained for tracking the position of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model with comparison to laser tracker during machining of Milling - 2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

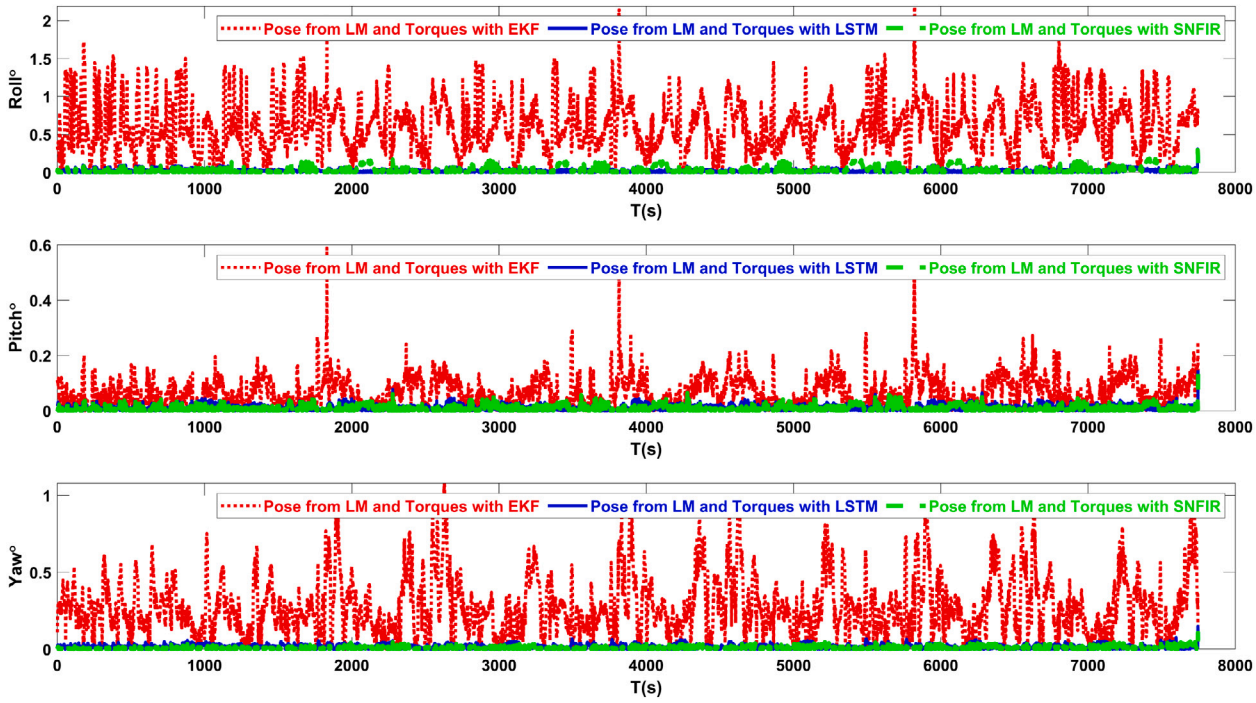


Fig. 15. Errors for tracking the orientation of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model during machining of Milling - 2.

Table 8
Pose tracking errors based on time series cross validation during machining, Milling - 2.

Algorithm	Validation error	Mean	Improvement percentile (%)	Standard deviation	Improvement percentile (%)
LM	ET (mm)	6.69		4.51	
	ER (°)	0.69		0.42	
LM and torques with EKF	ET (mm)	6.49	3.01	4.48	0.70
	ER (°)	0.68	1.56	0.40	3.43
LM and torques with LSTM	ET (mm)	4.43	33.65	4.25	5.73
	ER (°)	0.03	95.82	0.02	94.55
LM and torques with SNFIR	ET (mm)	2.49	62.76	2.23	50.64
	ER (°)	0.02	96.73	0.02	95.13

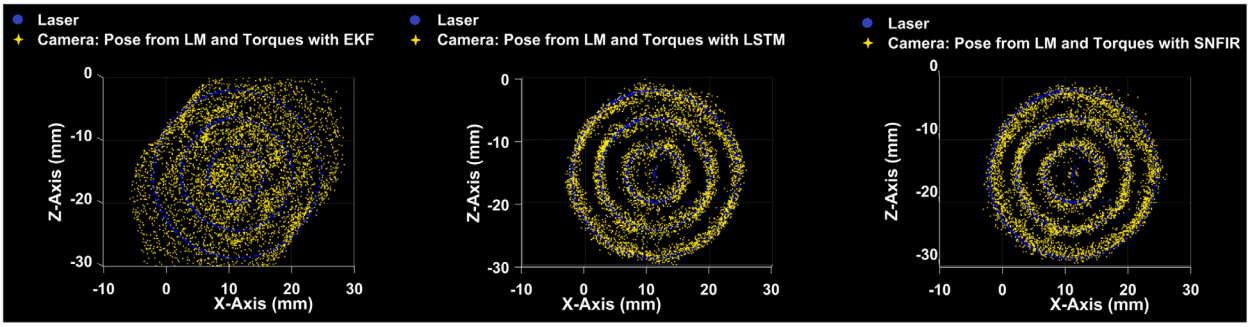


Fig. 16. Results obtained for tracking the position of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model with comparison to laser tracker during machining of Hole Milling. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

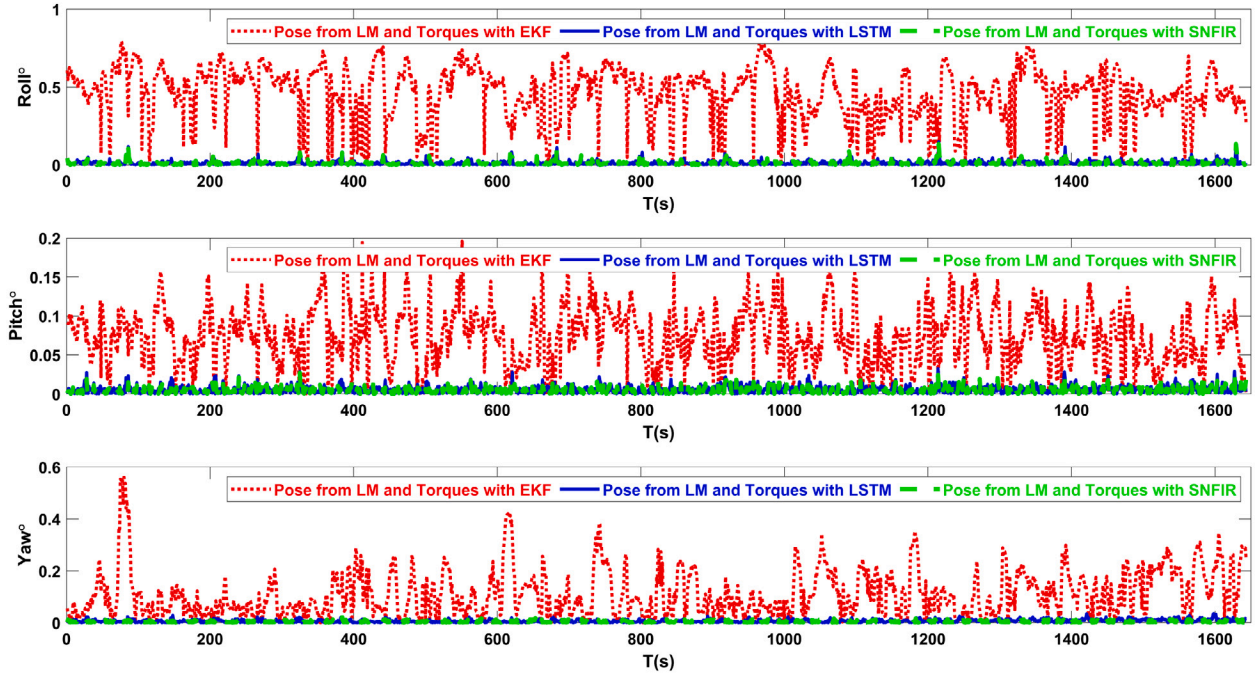


Fig. 17. Errors for tracking the orientation of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model during machining of Hole Milling.

Table 9
Pose tracking errors based on time series cross validation during machining, Hole Milling.

Algorithm	Validation error	Mean	Improvement percentile (%)	Standard deviation	Improvement percentile (%)
LM	ET (mm)	4.02		2.53	
	ER (°)	0.52		0.21	
LM and torques with EKF	ET (mm)	3.95	1.72	2.31	8.48
	ER (°)	0.49	5.15	0.20	2.13
LM and torques with LSTM	ET (mm)	1.27	68.37	1.02	59.83
	ER (°)	0.02	96.43	0.02	92.33
LM and torques with SNFIR	ET (mm)	1.55	61.49	1.29	49.14
	ER (°)	0.05	90.01	0.04	80.28

respectively when compared with the LM algorithm. Moreover, this proposed method was able to decrease the standard deviation of errors by 43.85% for position tracking and 93.25% for orientation tracking. As for the results obtained through SNFIR model for this machining process, it is seen that this method was able to improve the position tracking the most as it improved it by 66.98%. As for the orientation tracking errors, again the SNFIR model was able to reduce it the most by 95.20%. Moreover, this method was able to decrease the standard deviation of errors for position tracking by 58.35% and for orientation tracking by 95.01%.

As for the Milling - 2 machining process, the same pattern is observed as Milling - 1 machining. The results for this case are given in Fig. 14 for position trajectory and Fig. 15 for orientation tracking errors when the EKF, the proposed LSTM network and SNFIR model are utilized. In this machining case, the position tracking was improved by 3.01% and the orientation tracking was improved by 1.56% through the utilization of the EKF as tabulated in Table 8. The standard deviation of position and orientation tracking errors were also reduced, albeit a little, through the utilization of the EKF method by 0.7% and 3.43%, respectively. The proposed LSTM network was able to further improve the

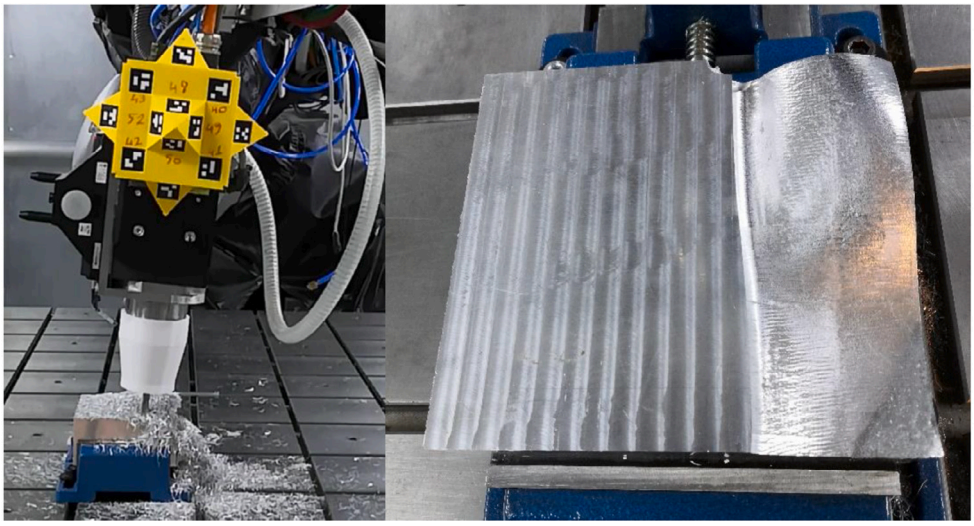


Fig. 18. Free form machined part, Milling - Roll Axis, with change in cutting tool's orientation in Roll axis only.

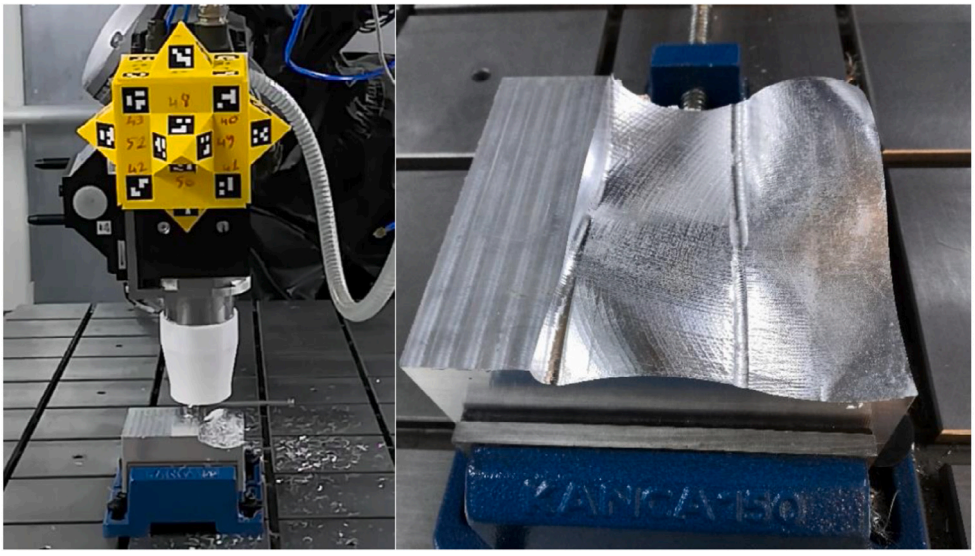


Fig. 19. Free form machined part, Milling - Yaw Axis, with change in cutting tool's orientation in Yaw axis only.

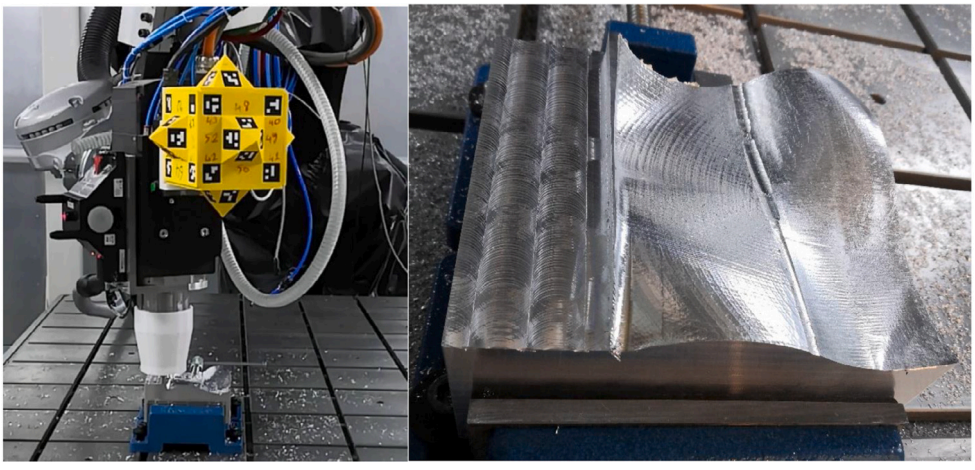


Fig. 20. Free form machined part, Milling - Pitch Axis, with change in cutting tool's orientation in Pitch axis only.

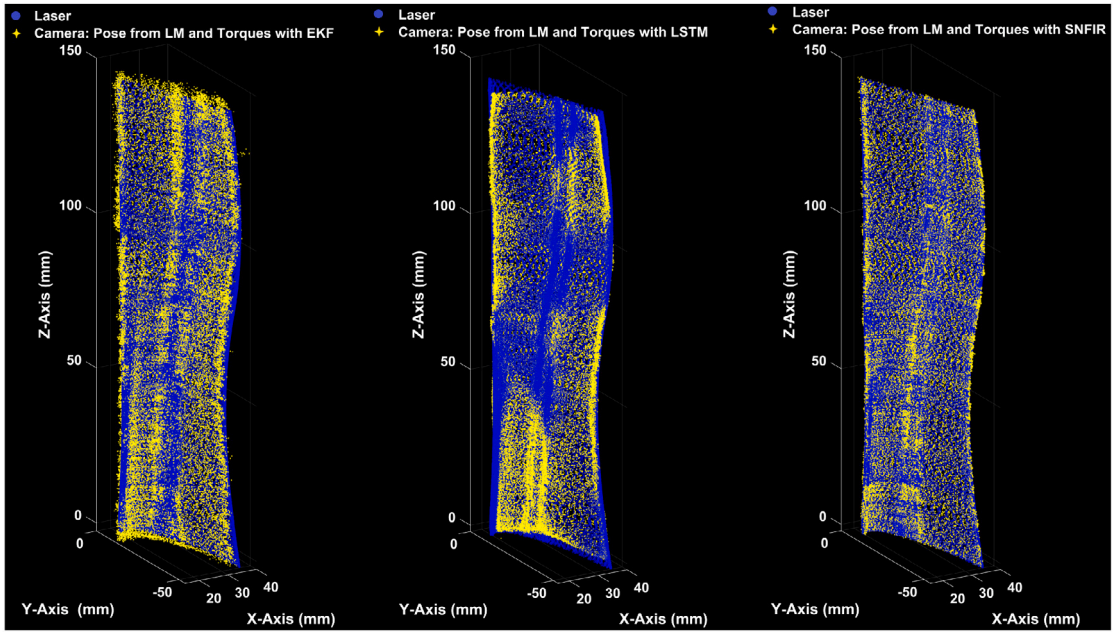


Fig. 21. Results obtained for tracking the position of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model with comparison to laser tracker during machining of Milling - Roll Axis.

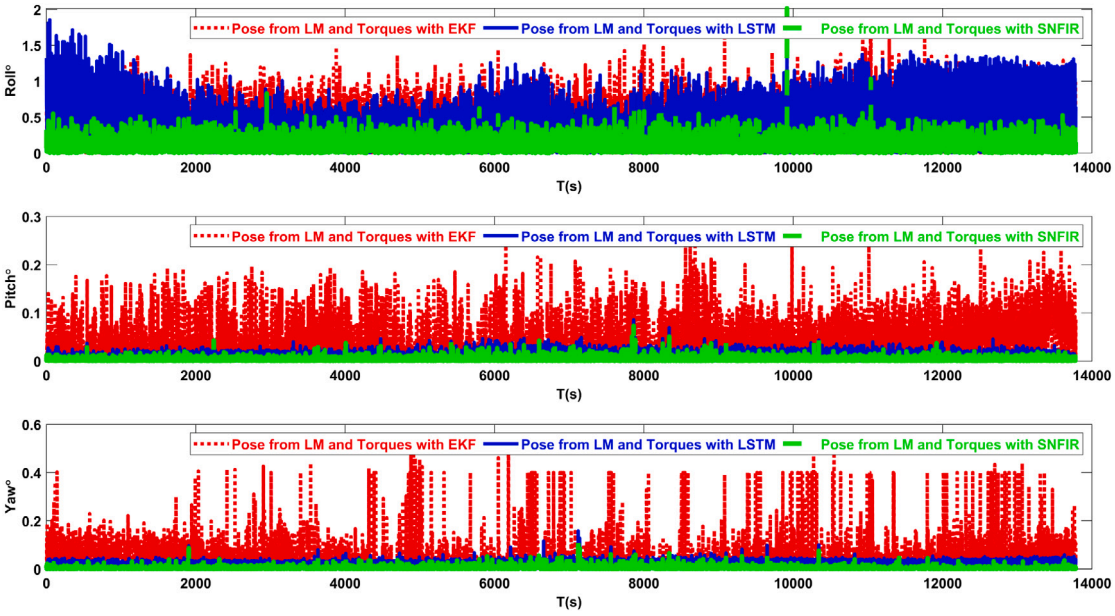


Fig. 22. Errors for tracking the orientation of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model during machining of Milling - Roll Axis.

Table 10
Pose tracking errors based on time series cross validation during machining, Milling - Roll Axis.

Algorithm	Validation error	Mean	Improvement percentile (%)	Standard deviation	Improvement percentile (%)
LM	ET (mm) ER (°)	2.28 0.22		1.80 0.23	
LM and torques with EKF	ET (mm) ER (°)	2.16 0.37	5.51 −65.29	2.02 0.36	−11.93 −53.02
LM and torques with LSTM	ET (mm) ER (°)	1.65 0.36	28.02 −62.87	1.91 0.30	−5.93 −26.89
LM and torques with SNFIR	ET (mm) ER (°)	0.59 0.12	74.23 45.88	0.48 0.13	73.36 45.39

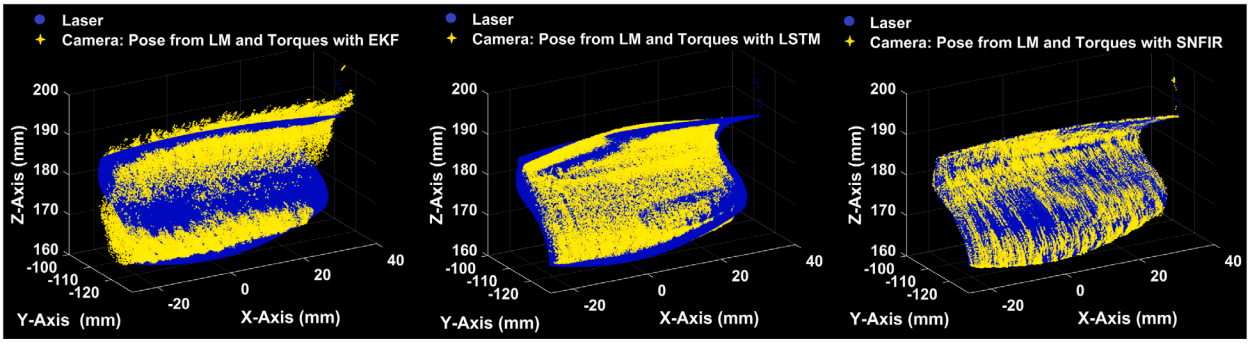


Fig. 23. Results obtained for tracking the position of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model with comparison to laser tracker during machining of Milling - Yaw Axis.

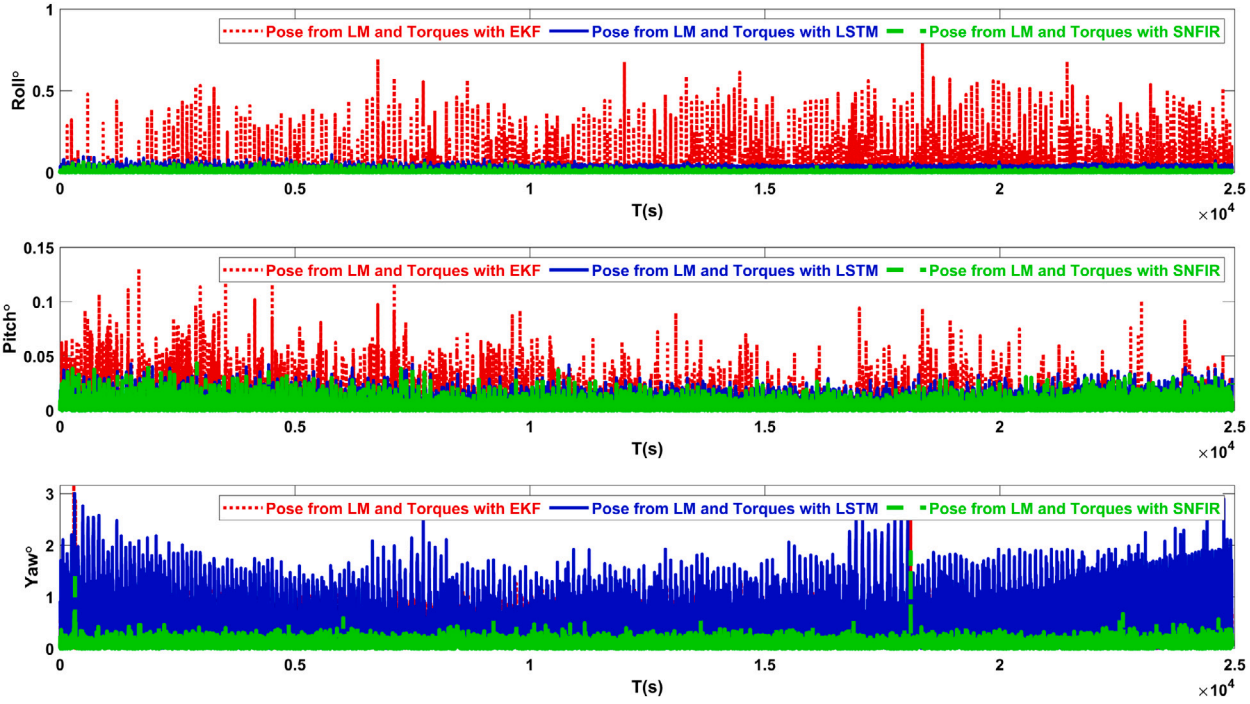


Fig. 24. Errors for tracking the orientation of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model during machining of Milling - Yaw Axis.

Table 11
Pose tracking errors based on time series cross validation during machining, Milling - Yaw Axis.

Algorithm	Validation error	Mean	Improvement percentile (%)	Standard deviation	Improvement percentile (%)
LM	ET (mm)	6.16		2.51	
	ER (°)	0.68		0.45	
LM and torques with EKF	ET (mm)	5.90	4.23	2.28	9.33
	ER (°)	0.65	4.55	0.43	3.82
LM and torques with LSTM	ET (mm)	1.49	75.84	1.60	36.21
	ER (°)	0.57	16.82	0.44	3.27
LM and torques with SNFIR	ET (mm)	0.45	92.69	0.37	85.17
	ER (°)	0.10	85.70	0.11	74.53

position tracking by 33.65% and the orientation tracking by 95.82%. Moreover, the standard deviation of errors were reduced by 5.73% for position tracking and 94.55% for orientation tracking. As for the results obtained through the SNFIR model for this machining process, this method was able to improve the position tracking by 62.76% which is twice the improvement when compare with the LSTM approach. As for the orientation tracking, this method was able to improve it by 96.73%. Moreover, the standard deviation of the errors were decreased by 50.64% for position tracking and by 95.13% for orientation tracking. During this machining process, only the SNFIR model was able to significantly reduce the standard deviation of position errors.

In Milling - 3 machining process, the significance of the achieved results is more observable since the machining was performed in a relatively small area as observed from Fig. 16. Moreover, the orientation tracking results are shown in Fig. 17. In this machining case, the EKF method slightly improved the position and orientation tracking by

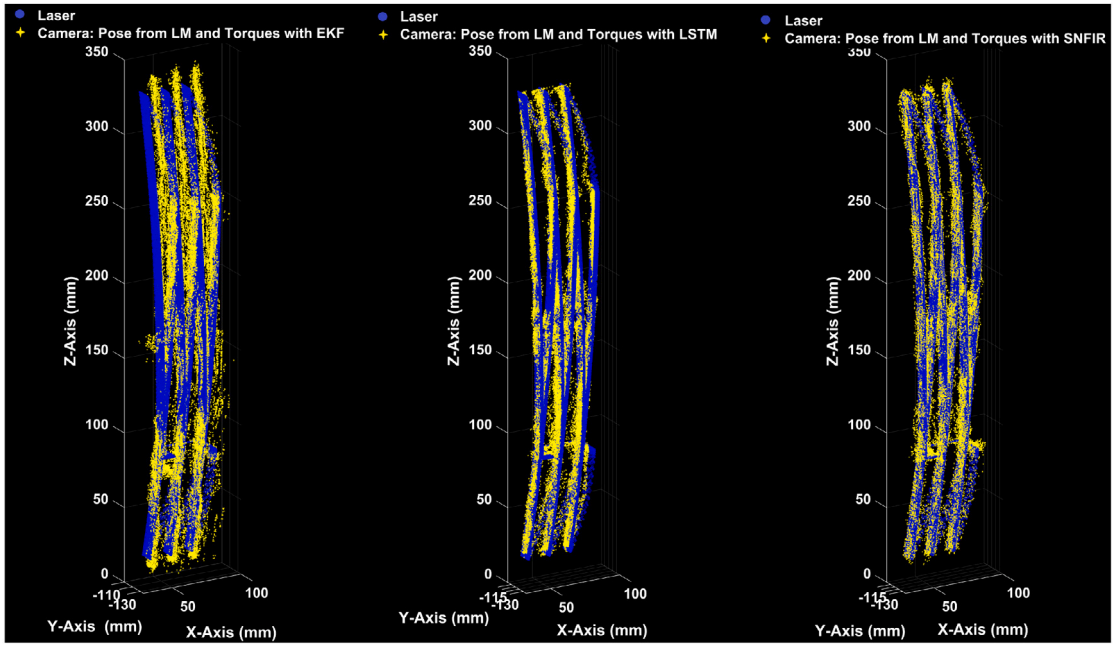


Fig. 25. Results obtained for tracking the position of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model with comparison to laser tracker during machining of Milling - Pitch Axis.

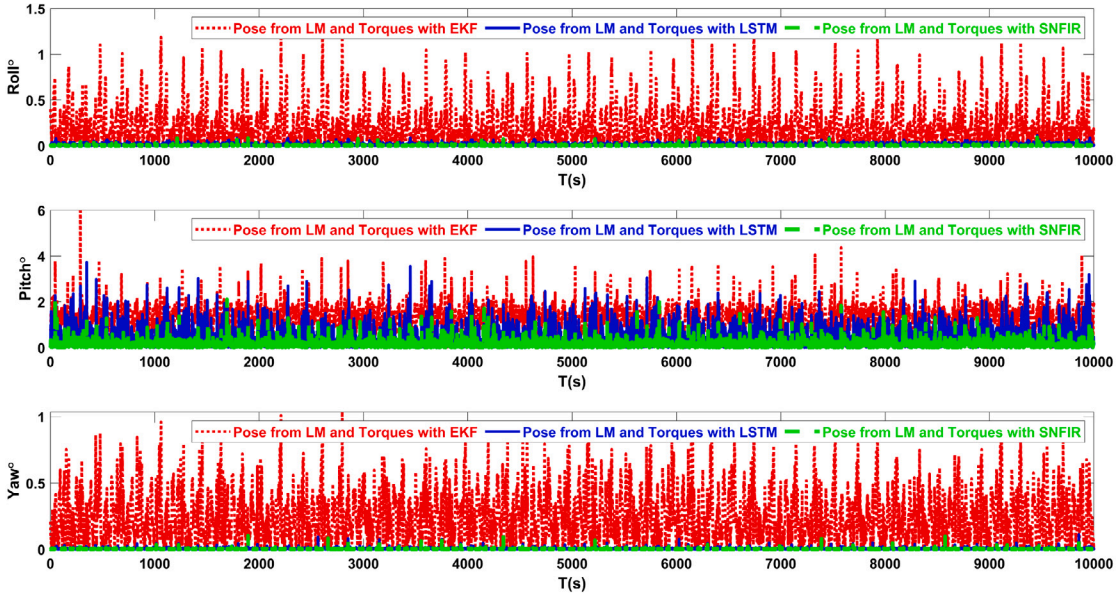


Fig. 26. Errors for tracking the orientation of the robot's cutting tool improved by the EKF, the proposed LSTM network and SNFIR model during machining of Milling - Pitch Axis.

Table 12
Pose tracking errors based on time series cross validation during machining, Milling - Pitch Axis.

Algorithm	Validation error	Mean	Improvement percentile (%)	Standard deviation	Improvement percentile (%)
LM	ET (mm) ER (°)	8.55 1.31		6.28 0.66	
LM and torques with EKF	ET (mm) ER (°)	7.99 1.29	6.63 1.68	6.14 0.64	2.26 4.46
LM and torques with LSTM	ET (mm) ER (°)	3.37 0.48	60.63 63.02	2.80 0.44	55.41 33.79
LM and torques with SNFIR	ET (mm) ER (°)	1.54 0.18	81.96 85.98	1.40 0.19	77.73 71.70

1.72% and 5.15%, respectively when compared with the pure LM based approach. The improvements for standard deviation of errors using this method were 8.48% and 2.13% for position and orientation tracking, respectively. In this machining process the LSTM proved to provide the best results, albeit by a small margin when compared with the SNFIR model. The proposed LSTM network improved the positioning tracking 68.37% and orientation tracking by 96.43%. Moreover, the standard deviation of errors were reduced by 59.83% for position tracking and by 92.33% for orientation tracking as shown in Table 9. As for the SNFIR model, this method was also able to significantly improve the position and orientation tracking by 61.49% and 90.01%, respectively. Moreover, the standard deviation of errors were decreased by 49.14% for position tracking and by 80.28% for orientation tracking.

As observed from these results of NAS 979 machining, the proposed LSTM network and SNFIR model are able to significantly improve the position tracking on average by 51.76% and 63.74%, respectively when compared with pure LM based pose estimation. Their effectiveness is especially apparent in the orientation tracking as they are able to improve it on average by 95.32% and 93.98%, respectively when all of the three stages of the NAS 979 machining are considered. Their capabilities are further validated when compared with the EKF method. In the NAS 979 machining the EKF was able to improve the position and orientation tracking on average by only 3.89% and 2.71%, respectively.

As for the evaluation of the proposed method for free form machining, as mentioned in Section 4.2, a machining process was performed in which the orientation of the cutting tool changed continuously for a single axis in three stages. The three stages of the free form machining process is given in Table 2. Initially the aluminum block was machined with the cutting tool's orientation continuously changing in Roll axis only as shown in Fig. 18. As observed this is a more complex shape when compared to the NAS 979 machining due to the changing orientation in addition to the changing position of the cutting tool of the robot. The results obtained through the EKF, the proposed LSTM network and SNFIR model for this case are shown in Fig. 21 for position tracking and Fig. 22 for orientation tracking. As observed from the results given in Table 10, the position tracking was improved by 5.51%, 28.02% and 74.23% when the EKF, LSTM network and the SNFIR model, respectively were utilized. However, neither the EKF nor the LSTM network were able to improve the orientation tracking in this machining case due to the complexity of the machining process. This is because in this case both the orientation and position of the cutting tool were continuously changing. Nonetheless, the SNFIR model was able to reduce the orientation errors by 45.88% while also improving the standard deviation of errors for position and orientation tracking. Whereas, both the EKF and the LSTM network were not able to improve the standard deviation of errors as well.

In the second stage of the free form machining, only the Yaw axis of the cutting tool was changing and the other were fixed as shown in Fig. 19. Figs. 23 and 24 show the trajectory tracking results and orientation tracking errors when compared with the laser tracker through utilization of the EKF, the proposed LSTM network and SNFIR model. In this machining scenario, position tracking was improved on average by 4.23%, 75.84% and 92.69% all the while providing 4.55%, 16.82% and 85.70% better orientation tracking when the EKF, the LSTM network and the SNFIR approaches were used, respectively. This can be seen from Table 11. Moreover, the standard deviation of errors were reduced on average by 9.33%, 36.21% and 85.17% for position tracking and by 3.82%, 3.27% and 74.53% for orientation tracking when the EKF, the LSTM network and the SNFIR approaches were utilized, respectively.

As for machining with changes only in Pitch axis as shown in Fig. 20, the obtained results through the EKF, the LSTM network and SNFIR model are shown in Figs. 25 and 26. From the results given in Table 12 the position tracking in this case were improved on average by 6.63%, 60.63% and 81.96% when the EKF, the LSTM network and the SNFIR model were utilized, respectively. This is in addition to improving the orientation tracking by 1.68%, 63.02% and 85.98%, respectively. Moreover, the standard deviation of errors were reduced on average by 2.26%, 55.41% and 77.73% for position tracking and by 4.46%, 33.79% and 71.70% for orientation tracking when the EKF, the LSTM network and the SNFIR approaches were used, respectively. It should be noted that, the improvements for milling around Yaw axis given in Table 11 are higher than the improvements for milling around Roll and Pitch axes given in Tables 10 and 12. This is due to multiple reasons. The first reason is the cutting speed in this case was halved when compared with the other cases due to the cutting direction thus resulting in acquisition of almost double amount of data when compared with the other cases. The second reason is cutting in this direction induced more vibration on the robot as observed from the roughness of the machined part in Fig. 19. Therefore, the proposed methods' capabilities to deal with noisy data resulting from vibrations and the utilization of the extra data due to the slower cutting speed resulted in this significant difference in improvement when compared with machining around Roll and Pitch axes.

From these results, it is clear that both of the proposed methods are able to significantly increase the accuracy and precision of the estimated pose by the LM based algorithm when the torque information is utilized during machining of standard parts such as NAS 979, in which the orientation of the cutting tool is fixed, and during free form machining, during which the orientation of the cutting tool continuously change. Moreover, it can be seen that the proposed SNFIR model provides 13.72%, 29.11%, 46.21%, 16.85% and 21.33% better position tracking and 1.49%, 0.91%, 108.75%, 68.88% and 22.96% better orientation tracking when compared with the results obtained through the proposed LSTM network in Milling-1, Milling-2, Milling - Roll Axis, Milling - Yaw Axis and Milling - Pitch Axis machining operations, respectively. The proposed LSTM network provides 6.88% and 6.42% better positioning and orientation tracking, respectively only in Milling-3 machining process when compared with the proposed SNFIR model. As for the improvement margins of the proposed SNFIR model over the proposed LSTM network for reducing standard deviation of errors, the same pattern and improvement ratios as the position and orientation tracking errors can be observed. Moreover, the SNFIR model provides parsimonious models as can be seen from the determined sparse coefficients for the six machining processes given in Tables 13 to 18. Based on the acquired models only $59.3\%, 53.6\%$ and 58.3% of the coefficients are active for position (δ_1, δ_2, and δ_3) and only $24.7\%, 25.7\%$ and 23.7% of the coefficients are active for orientation (δ_4, δ_5, and δ_6). This makes the model sparse in the space of possible functions thus determining only the fewest terms to accurately represent the data. Furthermore, such a method is very intuitive in that one can clearly see the coefficients defining the nonlinear system and thus provides more insight into the structure of the problem at hand.

Training a model using the proposed SNFIR model, such as for Milling - Yaw Axis, in MATLAB [43] takes only $17.85,\ 30.64$, and 47.96 s for $30\%,\ 50\%$, and 70% of the data containing 83153 samples. Whereas, training a model for the same machining case by using the proposed LSTM network requires about 2 h for each of the $30\%,\ 50\%$, and 70% of training cases. The training of the proposed LSTM networks requires the same time due to the training being performed using mini batches of same size and for the same number of iterations for all training cases. Besides these, both of the proposed methods can be used to improve the estimated pose in realtime, since the marker detection algorithm can work at 1000 Hz, the LM based pose estimation at 1000 Hz, the proposed LSTM network at 1000 Hz and the proposed SNFIR model at 3000 Hz for a single image. Therefore the total processing time for improved pose estimation is 0.003 s (333 Hz) for the proposed LSTM network and 0.00233 s (429 Hz) for the proposed SNFIR model. All of these computational times are summarized in Table 19. All of the computations performed in this work were realized on a workstation with an Intel Xeon E5-1650 CPU @ 3.5 GHz with 16 GB of RAM.

Table 13
The identified sparse coefficients using the proposed SNFIR approach when training a model for Milling - 1 machining with 70% of the data used for training.

	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6
1	50.81044	−4.07952	30.46101	−0.35948	−0.03799	0.140731
$\hat{T}_X(t)$	4.998681	−0.00623	−2.13869	0	0.000274	0
$\hat{T}_Y(t)$	2.098631	−0.20716	−0.74418	0.015715	0.000596	0.000193
$\hat{T}_Z(t)$	−1.97503	−0.03796	4.531231	0.000383	0.000331	−0.00062
$\hat{\alpha}(t)$	−14.3775	−2.27133	−12.9105	0.01014	−0.02903	0.003205
$\hat{\beta}(t)$	−15.6633	−8.83481	21.78823	0.13984	−0.09347	0.078892
$\hat{\gamma}(t)$	4.432111	7.527169	24.98618	−0.09493	0.016387	0.013964
$\tau_1(t)$	0	0.000332	0.000171	0	0	0
$\tau_2(t)$	0.000514	0.001832	−0.00569	0	0	0
$\tau_3(t)$	0.000759	0.000818	0.000195	0	0	0
$\tau_4(t)$	−0.02567	0.017069	−0.04121	0	0	0
$\tau_5(t)$	−0.01958	0.007156	−0.0159	0	0	0
$\tau_6(t)$	−0.02788	0.000381	−0.03452	0	0	0
$\hat{T}_X(t-1)$	−4.00915	0.009259	2.151808	0	−0.00027	0
$\hat{T}_Y(t-1)$	−1.31031	1.115836	1.276593	−0.01904	−0.0005	−0.0005
$\hat{T}_Z(t-1)$	1.943056	0.041218	−3.5802	−0.00034	−0.00032	0.000616
$\hat{\alpha}(t-1)$	4.992	3.760987	4.960615	0.00751	0.056805	−0.00575
$\hat{\beta}(t-1)$	13.12992	14.21632	−43.7384	−0.09864	0.064427	0.280344
$\hat{\gamma}(t-1)$	−4.11173	−7.29017	−31.8932	0.02287	−0.04227	0.201906
$\tau_1(t-1)$	0	−0.00023	−0.01111	0	0	0
$\tau_2(t-1)$	0.00863	−0.00838	0.017197	0	0	0
$\tau_3(t-1)$	0.006789	−0.00044	−0.0005	0	0	0
$\tau_4(t-1)$	0.042359	−0.02943	0.015806	0	0	0
$\tau_5(t-1)$	0.010707	−0.02323	−0.00542	0	0	0
$\tau_6(t-1)$	−0.08475	−0.03156	−0.06727	0	0	0
$\hat{T}_X(t)\hat{T}_X(t)$	−0.00175	0	−0.00236	0	0	0
$\hat{T}_X(t)\hat{T}_Y(t)$	0.159708	0.015872	−0.02569	0	0	0
$\hat{T}_X(t)\hat{T}_Z(t)$	−0.00864	0	−0.00606	0	0	0
$\hat{T}_X(t)\hat{\alpha}(t)$	0.30895	0.027156	−0.1394	−0.00045	−0.00033	−0.00018
$\hat{T}_X(t)\hat{\beta}(t)$	0.351206	0.011153	−0.54714	0.008137	0.001065	−0.00307
$\hat{T}_X(t)\hat{\gamma}(t)$	−1.05576	−0.16419	−0.19247	0.004583	0.001086	−0.00133
$\hat{T}_X(t)\tau_1(t)$	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{T}_X(t)\tau_6(t)$	0.000207	0	−0.00073	0	0	0
$\hat{T}_X(t)\hat{T}_X(t-1)$	0.002308	0	0.004367	0	0	0
$\hat{T}_X(t)\hat{T}_Y(t-1)$	−0.12409	−0.01566	0.014087	0	0	0
$\hat{T}_X(t)\hat{T}_Z(t-1)$	0.009206	0	0.006082	0	0	0
$\hat{T}_X(t)\hat{\alpha}(t-1)$	−0.39187	−0.01495	−0.00633	−0.00044	0	0.000733
$\hat{T}_X(t)\hat{\beta}(t-1)$	−0.56725	−0.05884	−0.27808	−0.00745	−0.00125	0.001568
$\hat{T}_X(t)\hat{\gamma}(t-1)$	0.709376	0.148568	0.187405	−0.00287	−0.00052	−0.00027
$\hat{T}_X(t)\tau_1(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_2(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_3(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_4(t-1)$	0	0	−0.00084	0	0	0
$\hat{T}_X(t)\tau_5(t-1)$	0.000517	0	0	0	0	0
$\hat{T}_X(t)\tau_6(t-1)$	0	0	−0.00092	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{\gamma}(t)\tau_6(t-1)$	0.035247	0.017498	0.023539	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\tau_6(t-1)\tau_6(t-1)$	0	0	0	0	0	0

Table 14
The identified sparse coefficients using the proposed SNFIR approach when training a model for Milling - 2 machining with 70% of the data used for training.

	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6
1	−36.7639	26.66856	5.070741	−0.15796	−0.05615	0.193898
$\hat{T}_X(t)$	3.030323	−0.05191	−1.2256	0.001235	0	−0.00154
$\hat{T}_Y(t)$	−3.27889	1.899064	−1.72056	0.0008	0.000378	0.000158
$\hat{T}_Z(t)$	−1.01966	−0.0177	5.190509	0	−0.00014	−0.0004
$\hat{\alpha}(t)$	−1.86681	1.76286	−1.40419	−0.04465	−0.00051	0.044572
$\hat{\beta}(t)$	−6.8388	9.60194	67.16018	−0.17435	0.019413	0.089696
$\hat{\gamma}(t)$	3.003367	−4.48484	12.22481	0.025074	−0.00679	−0.06207
$\tau_1(t)$	0.000329	0.004682	0	0	0	0
$\tau_2(t)$	0	0.000551	−0.00034	0	0	0
$\tau_3(t)$	0.000347	0.000209	0	0	0	0
$\tau_4(t)$	−0.03524	−0.01091	−0.02725	0	0	0
$\tau_5(t)$	0.017975	−0.00624	0.037524	0	0	0
$\tau_6(t)$	0.003659	0.007008	−0.05669	0.00022	0	0
$\hat{T}_X(t-1)$	−1.98201	0.051307	1.270439	−0.00123	0	0.001562
$\hat{T}_Y(t-1)$	2.8288	−0.38695	1.983513	−0.00072	−0.00041	0
$\hat{T}_Z(t-1)$	0.994534	0.024426	−4.26548	0	0.000142	0.000395
$\hat{\alpha}(t-1)$	−0.1204	−0.89497	−3.781	0.00865	0.014104	−0.00628
$\hat{\beta}(t-1)$	39.89597	−2.6407	−44.931	0.089718	0.063846	−0.04632
$\hat{\gamma}(t-1)$	5.597864	−0.62206	−1.81259	−0.02817	0.028663	0.077263
$\tau_1(t-1)$	−0.00986	−0.00092	0	0	0	0
$\tau_2(t-1)$	0.000675	−0.00513	0.000672	0	0	0
$\tau_3(t-1)$	−0.00891	−0.00053	−0.0007	0	0	0
$\tau_4(t-1)$	−0.01003	0	−0.07064	0	0	0
$\tau_5(t-1)$	0.001964	−0.00136	−0.02142	0	0	0
$\tau_6(t-1)$	−0.13542	−0.02117	−0.14695	−0.00018	0	0
$\hat{T}_X(t)\hat{T}_X(t)$	0.00573	0.001412	−0.00431	0	0	0
$\hat{T}_X(t)\hat{T}_Y(t)$	−0.10322	−0.0002	−0.15963	0	0	0
$\hat{T}_X(t)\hat{T}_Z(t)$	−0.00192	0	0.011295	0	0	0
$\hat{T}_X(t)\hat{\alpha}(t)$	−0.21505	0.006102	−0.1536	0.000576	0.000147	−0.00057
$\hat{T}_X(t)\hat{\beta}(t)$	−0.85316	0.135309	−0.50368	0.009078	0.001896	−0.01304
$\hat{T}_X(t)\hat{\gamma}(t)$	0.183537	0.09824	0.221657	0.00352	−0.00022	0
$\hat{T}_X(t)\tau_1(t)$	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{T}_X(t)\tau_6(t)$	0.000591	−0.00039	−0.00178	0	0	0
$\hat{T}_X(t)\hat{T}_X(t-1)$	−0.01251	−0.00275	0.008436	0	0	0
$\hat{T}_X(t)\hat{T}_Y(t-1)$	0.131238	0	0.151216	0	0	0
$\hat{T}_X(t)\hat{T}_Z(t-1)$	0.002263	0	−0.01077	0	0	0
$\hat{T}_X(t)\hat{\alpha}(t-1)$	0.097131	0.022456	0.138031	0	−0.00057	0
$\hat{T}_X(t)\hat{\beta}(t-1)$	0.76202	−0.02055	0.967722	−0.00059	−0.00264	−0.00055
$\hat{T}_X(t)\hat{\gamma}(t-1)$	−0.31703	−0.10609	−0.22951	−0.00432	0.00142	0
$\hat{T}_X(t)\tau_1(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_2(t-1)$	0	0	0.000212	0	0	0
$\hat{T}_X(t)\tau_3(t-1)$	−0.00035	0	0	0	0	0
$\hat{T}_X(t)\tau_4(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_5(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_6(t-1)$	0.00015	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{\gamma}(t)\tau_6(t-1)$	0.030448	0.004859	0.005283	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\tau_6(t-1)\tau_6(t-1)$	0	0	0	0	0	0

Table 15
The identified sparse coefficients using the proposed SNFIR approach when training a model for Milling - 3 machining with 70% of the data used for training.

	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6
1	35.98508	4.710585	26.66388	−0.09417	−0.07403	0.184702
$\hat{T}_X(t)$	−0.84261	0.284053	−0.23018	−0.01932	0.003767	0.000323
$\hat{T}_Y(t)$	6.265269	−2.00642	0.382623	0.035019	−0.00589	−0.01865
$\hat{T}_Z(t)$	−0.57948	0.245835	1.041346	−0.01751	0.001292	−0.0023
$\hat{\alpha}(t)$	−12.4434	2.577153	−12.2943	−0.13592	0.001381	0.043498
$\hat{\beta}(t)$	2.662144	3.455102	−24.1434	−0.67456	0.123272	−0.01276
$\hat{\gamma}(t)$	−36.5684	6.623401	−0.61297	−0.48748	0.106531	0.002342
$\tau_1(t)$	−0.00345	0	0.002717	0	0	0
$\tau_2(t)$	0.004953	0.00017	−0.00079	0	0	0
$\tau_3(t)$	0	0.000921	−0.00105	0	0	0
$\tau_4(t)$	0.026072	−0.00027	−0.01549	0	0	0
$\tau_5(t)$	0.003078	−0.00028	−0.00284	0	0	0
$\tau_6(t)$	−0.00684	0.003717	0.014451	0	0	0
$\hat{T}_X(t-1)$	2.014586	−0.37016	0.057721	0.019094	0	0
$\hat{T}_Y(t-1)$	−5.78862	3.107507	−0.03459	−0.04016	0.005749	0.0186
$\hat{T}_Z(t-1)$	0.525496	−0.30031	0.320852	0.010893	0	0
$\hat{\alpha}(t-1)$	6.115568	−3.87272	6.150183	0.090045	−0.02886	−0.00251
$\hat{\beta}(t-1)$	−0.19492	−2.02127	38.79309	0.031847	−0.15013	0.244929
$\hat{\gamma}(t-1)$	36.03664	−11.2465	15.09552	0.501512	−0.13446	−0.0791
$\tau_1(t-1)$	−0.00742	0	0.006751	0	0	0
$\tau_2(t-1)$	−0.0024	0.000362	−0.00046	0	0	0
$\tau_3(t-1)$	0.002422	−0.00047	0	0	0	0
$\tau_4(t-1)$	0.004897	0.004723	0.003681	0	0	0
$\tau_5(t-1)$	0.003941	0.00332	0.007911	0	0	0
$\tau_6(t-1)$	0.028233	−0.00191	0.006804	0.00035	0	0
$\hat{T}_X(t)\hat{T}_X(t)$	−0.07405	−0.01948	0.001888	0	0	0
$\hat{T}_X(t)\hat{T}_Y(t)$	0.400259	0.151736	−0.04301	−0.00518	0.00032	−0.00117
$\hat{T}_X(t)\hat{T}_Z(t)$	−0.00877	−0.05732	−0.01435	−0.00097	−0.00035	0
$\hat{T}_X(t)\hat{\alpha}(t)$	0.139337	0.303742	−0.2708	0.00836	0.00159	−0.00245
$\hat{T}_X(t)\hat{\beta}(t)$	−0.18409	0.744688	−1.78177	0.030004	0.001979	−0.02551
$\hat{T}_X(t)\hat{\gamma}(t)$	−1.03799	−0.68302	0.153536	−0.01709	0.003161	0.013949
$\hat{T}_X(t)\tau_1(t)$	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{T}_X(t)\tau_6(t)$	−0.00372	0.000353	−0.00126	0	0	0
$\hat{T}_X(t)\hat{T}_X(t-1)$	0.137345	0.040817	0.020631	0.000266	0	0
$\hat{T}_X(t)\hat{T}_Y(t-1)$	−0.40746	−0.14671	0.047122	0.005001	−0.00029	0.001189
$\hat{T}_X(t)\hat{T}_Z(t-1)$	−0.01265	0.051574	0.008873	0.001022	0.000356	0
$\hat{T}_X(t)\hat{\alpha}(t-1)$	−0.31151	−0.44474	0.009102	−0.01655	−0.00198	−0.00023
$\hat{T}_X(t)\hat{\beta}(t-1)$	−1.65674	−1.54565	−0.09351	−0.05166	−0.00583	0.011435
$\hat{T}_X(t)\hat{\gamma}(t-1)$	0.554167	0.760091	−0.31157	0.010555	−0.00398	−0.01334
$\hat{T}_X(t)\tau_1(t-1)$	0	0	0.000385	0	0	0
$\hat{T}_X(t)\tau_2(t-1)$	0	0.000138	0	0	0	0
$\hat{T}_X(t)\tau_3(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_4(t-1)$	−0.00152	0	0.002046	0	0	0
$\hat{T}_X(t)\tau_5(t-1)$	0	0.00043	0	0	0	0
$\hat{T}_X(t)\tau_6(t-1)$	0	0	0.002031	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{\gamma}(t)\tau_6(t-1)$	−0.01999	−0.00094	0.084561	−0.00074	0	0.000276
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\tau_6(t-1)\tau_6(t-1)$	0	0	0	0	0	0

Table 16
The identified sparse coefficients using the proposed SNFIR approach when training a model for Milling - Roll machining with 70% of the data used for training.

	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6
1	17.2827	−4.87282	19.69081	−2.59295	−0.04324	−0.04
$\hat{T}_X(t)$	0.131144	−0.14704	−0.82731	0.018014	0.001276	0
$\hat{T}_Y(t)$	0.028254	0.144746	0.069002	−0.03412	−0.00041	0.000249
$\hat{T}_Z(t)$	0.023595	0.17308	0.482064	−0.06529	0.004605	−0.01367
$\hat{\alpha}(t)$	−0.85053	1.46338	1.064643	0.236572	−0.011	−0.00085
$\hat{\beta}(t)$	−0.94596	−0.81389	0.081276	−0.00788	−0.02677	0.09626
$\hat{\gamma}(t)$	−0.87598	5.354241	−3.93494	−0.03974	0.027424	−0.01126
$\tau_1(t)$	0	0.000149	0	−0.00034	0	0
$\tau_2(t)$	0	0.000135	0	0	0	0
$\tau_3(t)$	0	0	0	0	0	0
$\tau_4(t)$	0.01542	−0.07391	−0.00081	0.007156	0	0
$\tau_5(t)$	0.000288	−0.00594	0.000308	0	0	0
$\tau_6(t)$	−0.00079	−0.01162	0.006712	0.000508	0	0
$\hat{T}_X(t-1)$	0.576524	0.223157	−0.30778	0.149331	0.000237	0.001549
$\hat{T}_Y(t-1)$	−0.03156	0.887364	0.024139	0.009295	0.000527	−0.00056
$\hat{T}_Z(t-1)$	−0.01384	−0.14982	0.470192	0.069842	−0.00444	0.014125
$\hat{\alpha}(t-1)$	1.680145	−2.18345	0.11087	0.482231	0.004188	−0.00734
$\hat{\beta}(t-1)$	−1.97245	0.612182	0.232835	−1.64217	0.077286	−0.00895
$\hat{\gamma}(t-1)$	4.765243	−0.49495	−1.32294	1.677128	0.031393	0.063424
$\tau_1(t-1)$	0	−0.00099	0	0	0	0
$\tau_2(t-1)$	0	0	0	0	0	0
$\tau_3(t-1)$	0	−0.00057	0	0	0	0
$\tau_4(t-1)$	0.000163	0.018642	0.000446	−0.00023	0	0
$\tau_5(t-1)$	−0.00014	−0.00202	−0.00024	0	0	0
$\tau_6(t-1)$	−0.01804	0.014035	−0.00047	−0.00109	0	0
$\hat{T}_X(t)\hat{T}_X(t)$	−0.00634	0.012041	0.01261	−0.00255	0	0
$\hat{T}_X(t)\hat{T}_Y(t)$	−0.00215	0.013128	−0.00567	−0.00203	0	0
$\hat{T}_X(t)\hat{T}_Z(t)$	0.001701	−0.01619	−0.00628	0.005421	0	0
$\hat{T}_X(t)\hat{\alpha}(t)$	0.068758	−0.08767	−0.03894	0.01767	0	−0.00016
$\hat{T}_X(t)\hat{\beta}(t)$	0.022107	0.098707	0.054795	−0.01445	0.000601	−0.00233
$\hat{T}_X(t)\hat{\gamma}(t)$	−0.00332	−0.03138	0.104544	0.007535	−0.0003	0
$\hat{T}_X(t)\tau_1(t)$	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{T}_X(t)\tau_6(t)$	0	0	0	0	0	0
$\hat{T}_X(t)\hat{T}_X(t-1)$	0.007848	−0.01999	−0.00217	0.004145	0	0
$\hat{T}_X(t)\hat{T}_Y(t-1)$	0.001275	−0.019	0.000335	0.001212	0	0
$\hat{T}_X(t)\hat{T}_Z(t-1)$	−0.00201	0.014995	0.006199	−0.00569	0	0
$\hat{T}_X(t)\hat{\alpha}(t-1)$	−0.06262	0.076792	0.001035	−0.01948	0	0
$\hat{T}_X(t)\hat{\beta}(t-1)$	0.038941	0	−0.06803	0.043112	−0.00275	0.00064
$\hat{T}_X(t)\hat{\gamma}(t-1)$	0.001586	0	0.060571	−0.01264	0	0.000145
$\hat{T}_X(t)\tau_1(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_2(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_3(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_4(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_5(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_6(t-1)$	0.00056	−0.00077	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{\gamma}(t)\tau_6(t-1)$	0.001254	−0.00557	−0.00083	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\tau_6(t-1)\tau_6(t-1)$	0	0	0	0	0	0

Table 17
The identified sparse coefficients using the proposed SNFIR approach when training a model for Milling - Yaw machining with 70% of the data used for training.

	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6
1	0	0	0	0	0	0
$\hat{T}_X(t)$	−0.35108	−20.3415	2.42634	−0.10775	−0.05849	−4.5507
$\hat{T}_Y(t)$	−0.69775	8.52915	0.420679	0.000844	0.001524	−0.73794
$\hat{T}_Z(t)$	−0.7304	4.205359	−3.83324	0.002849	0.00173	0.087512
$\hat{\alpha}(t)$	−6.2499	−99.5261	4.371383	−0.43389	−0.28554	−27.5116
$\hat{\beta}(t)$	−15.9527	32.06225	57.9087	0.867616	0.955307	−20.9048
$\hat{\gamma}(t)$	0.68857	−22.5816	11.94872	−0.02683	−0.08151	−2.18192
$\tau_1(t)$	0	0.035395	−0.00031	0	0	0.000536
$\tau_2(t)$	0	0.026075	−0.04917	0	0	0
$\tau_3(t)$	−0.00016	0.046025	0.000548	0	0	−0.00033
$\tau_4(t)$	0.000717	−0.08754	−0.05849	0	0	0.001383
$\tau_5(t)$	−0.0328	0.008543	0.01088	0	0	0.00141
$\tau_6(t)$	0.076931	0.087381	0.332139	0	0	−0.0015
$\hat{T}_X(t-1)$	1.253278	20.76929	−2.63416	0.108902	0.05857	4.663169
$\hat{T}_Y(t-1)$	−1.00755	−8.92763	0.666755	−0.00037	−0.00145	−0.02531
$\hat{T}_Z(t-1)$	−0.20966	−5.18761	6.582951	0.001155	0	−0.59017
$\hat{\alpha}(t-1)$	4.591463	82.44863	−0.0703	0.577319	0.325894	22.73316
$\hat{\beta}(t-1)$	−13.1572	63.85863	−124.186	0.579431	0.178339	27.37976
$\hat{\gamma}(t-1)$	−1.53281	27.07562	−7.46509	0.033164	0.098767	2.784019
$\tau_1(t-1)$	0	−0.03952	0.000558	0	0	−0.00068
$\tau_2(t-1)$	0	−0.00145	0	0	0	0.000146
$\tau_3(t-1)$	0.000135	−0.07826	−0.00067	0	0	0.000423
$\tau_4(t-1)$	0.000662	0.248286	−0.16231	0	0	−0.02137
$\tau_5(t-1)$	0	−0.08604	−0.04716	0	0	−0.00157
$\tau_6(t-1)$	0.001151	0.025913	0.039375	0	0	−0.03723
$\hat{T}_X(t)\hat{T}_X(t)$	−0.01034	−0.05418	0.006775	0	0.000116	−0.00604
$\hat{T}_X(t)\hat{T}_Y(t)$	0.013535	−0.01177	−0.0165	−0.00036	0	0.008288
$\hat{T}_X(t)\hat{T}_Z(t)$	0.004825	0.025889	−0.01091	0.000369	0.000195	0.011222
$\hat{T}_X(t)\hat{\alpha}(t)$	−0.07795	−0.4857	0.005189	0	0.000527	−0.0387
$\hat{T}_X(t)\hat{\beta}(t)$	−0.05968	−1.67705	−0.2635	−0.0042	−0.00038	0.011562
$\hat{T}_X(t)\hat{\gamma}(t)$	−0.01555	−0.30327	0.037713	0	0.000646	−0.02443
$\hat{T}_X(t)\tau_1(t)$	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{T}_X(t)\tau_6(t)$	−0.00039	−0.00568	0.001016	0	0	−0.00162
$\hat{T}_X(t)\hat{T}_X(t-1)$	0.020122	0.106613	−0.00978	0	−0.00031	0.009345
$\hat{T}_X(t)\hat{T}_Y(t-1)$	−0.01785	0.051269	−0.00405	0	0.00034	0.002106
$\hat{T}_X(t)\hat{T}_Z(t-1)$	−0.00291	0.122478	−0.01666	0	0.000345	0.022029
$\hat{T}_X(t)\hat{\alpha}(t-1)$	0.098617	0.892187	−0.02991	0.000495	0.000346	0.094601
$\hat{T}_X(t)\hat{\beta}(t-1)$	0.173586	−0.52349	0.31653	0.008992	0.00696	−0.21217
$\hat{T}_X(t)\hat{\gamma}(t-1)$	0.01221	0.535027	−0.09112	0	0.000198	0.072404
$\hat{T}_X(t)\tau_1(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_2(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_3(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_4(t-1)$	−0.0004	0.000746	0.000209	0	0	−0.00072
$\hat{T}_X(t)\tau_5(t-1)$	−0.0005	−0.00085	0	0	0	0
$\hat{T}_X(t)\tau_6(t-1)$	0	−0.00116	0.000679	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{\gamma}(t)\tau_6(t-1)$	0.000113	−0.00521	0	0	0	−0.00043
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\tau_6(t-1)\tau_6(t-1)$	0	0	0	0	0	0

Table 18
The identified sparse coefficients using the proposed SNFIR approach when training a model for Milling - Pitch machining with 70% of the data used for training.

	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6
1	62.2912	−10.2331	142.8483	0.62018	0.479671	−0.05362
$\hat{T}_X(t)$	2.205667	−0.02888	−0.06233	0.000955	−0.1247	0
$\hat{T}_Y(t)$	0.255031	1.755511	−0.92544	−0.00258	0.097686	0.001849
$\hat{T}_Z(t)$	0.727959	0.123955	1.484754	0.000235	−0.00892	−0.00051
$\hat{\alpha}(t)$	−39.5263	11.92615	−9.72322	0.136162	9.303861	−0.16369
$\hat{\beta}(t)$	7.339619	0.303451	−1.51745	0.005203	0.615204	0.001775
$\hat{\gamma}(t)$	−29.9425	−2.38417	7.015326	0.075822	6.706711	−0.15083
$\tau_1(t)$	−0.0091	0	−0.00926	0	−0.00055	0
$\tau_2(t)$	−0.00575	0	0	0	0	0
$\tau_3(t)$	0.002847	−0.0002	0.00708	0	0	0
$\tau_4(t)$	0.037332	0.000207	0.144231	0	0.005042	0
$\tau_5(t)$	−0.00542	0	0.01874	0	0.000409	0
$\tau_6(t)$	0.089242	−0.00048	0.214967	0	−0.00104	0
$\hat{T}_X(t-1)$	−1.04328	0.013579	−0.07984	−0.00096	0.09219	0
$\hat{T}_Y(t-1)$	−0.12179	−0.76796	2.839553	0.002852	0.063992	−0.00154
$\hat{T}_Z(t-1)$	−1.40536	−0.09497	−0.37259	0	0.140374	0.0003
$\hat{\alpha}(t-1)$	−14.8415	−2.55625	−1.01187	−0.06653	1.539167	−0.08189
$\hat{\beta}(t-1)$	−12.2632	−0.15459	2.532216	0.003123	1.27282	−0.00377
$\hat{\gamma}(t-1)$	−15.0124	2.508672	−7.80459	−0.07174	0.419181	−0.03202
$\tau_1(t-1)$	0.042113	−0.00015	0.051087	0	−0.00549	0
$\tau_2(t-1)$	0.012652	−0.0004	−0.00175	0	−0.00036	0
$\tau_3(t-1)$	0.000896	0	−0.00472	0	0.001019	0
$\tau_4(t-1)$	−0.05946	−0.00414	−0.0514	0	0.005402	0
$\tau_5(t-1)$	−0.00222	0.00051	−0.03098	0	−0.00023	0
$\tau_6(t-1)$	−0.15276	−0.00081	0.028221	0	−0.00252	0
$\hat{T}_X(t)\hat{T}_X(t)$	0.004478	−0.00042	0	0	−0.00013	0
$\hat{T}_X(t)\hat{T}_Y(t)$	−0.03932	−0.00024	−0.02274	0	0.007686	0
$\hat{T}_X(t)\hat{T}_Z(t)$	−0.0028	−0.00074	−0.01183	0	0.001321	0
$\hat{T}_X(t)\hat{\alpha}(t)$	−0.4814	−0.00137	−0.38215	0	0.100345	0
$\hat{T}_X(t)\hat{\beta}(t)$	0.025786	−0.00379	−0.03059	0	0.004744	0
$\hat{T}_X(t)\hat{\gamma}(t)$	−0.14047	0	−0.05713	0.00089	−0.02435	0
$\hat{T}_X(t)\tau_1(t)$	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{T}_X(t)\tau_6(t)$	0.00014	0	0	0	0	0
$\hat{T}_X(t)\hat{T}_X(t-1)$	−0.00781	0.000847	0.000397	0	0.00022	0
$\hat{T}_X(t)\hat{T}_Y(t-1)$	0.051408	0	0.020466	0	−0.00882	0
$\hat{T}_X(t)\hat{T}_Z(t-1)$	0.002542	0.000722	0.010061	0	−0.00127	0
$\hat{T}_X(t)\hat{\alpha}(t-1)$	0.382572	−0.00696	0.22976	0	−0.09958	0
$\hat{T}_X(t)\hat{\beta}(t-1)$	−0.03094	0.00346	0.023962	0	−0.00458	0
$\hat{T}_X(t)\hat{\gamma}(t-1)$	0.112772	−0.0077	−0.03503	−0.00118	0.004412	0
$\hat{T}_X(t)\tau_1(t-1)$	0	0	0.000184	0	0	0
$\hat{T}_X(t)\tau_2(t-1)$	0	0	−0.00017	0	0	0
$\hat{T}_X(t)\tau_3(t-1)$	0	0	0	0	0	0
$\hat{T}_X(t)\tau_4(t-1)$	0.001166	0	0.000212	0	0	0
$\hat{T}_X(t)\tau_5(t-1)$	−0.00057	0	0.000565	0	0	0
$\hat{T}_X(t)\tau_6(t-1)$	0.001075	0	0.000504	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{\gamma}(t)\tau_6(t-1)$	−0.03011	−0.00022	−0.01061	0	0.009352	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\tau_6(t-1)\tau_6(t-1)$	0	0	0	0	0	0

Table 19
Computational times for the tasks in the proposed methods.

Task	Computational time
Image acquisition	0.00266 s (375 Hz)
Marker detection	0.001 s (1000 Hz)
Pose estimation using LM	0.001 s (1000 Hz)
Pose improvement using EKF for a single sample	0.001 s (1000 Hz)
Pose improvement using LSTM for a single sample	0.001 s (1000 Hz)
Pose improvement using SNFIR for a single sample	0.00033 s (3000 Hz)
LSTM training using 30%, 50% and 70% of milling around Yaw axis dataset	2 h
SNFIR training using 30%, 50% and 70% of milling around Yaw axis dataset	17.85, 30.64 and 47.96 s

5. Conclusion

In this work a monocular machine vision based pose estimation system was developed for robotic machining. First, a target to be tracked by the camera system was designed with fiducial markers which guarantees the detection of at least two non-parallel markers from ±90° in all directions of the camera's view. Afterwards, an optimization approach was proposed to minimize the errors in determination of the corner points of the fiducial markers in object frame so as to minimize the errors in vision based pose estimation. Moreover, to improve the pose estimated by the Levenberg–Marquardt (LM) based algorithm in realtime, two data driven approaches were proposed which can take the dynamics due to robotic machining into account through utilization of the torques induced at the robot's joints. Both of the proposed methods utilized the pose estimated by Levenberg–Marquardt (LM) in addition to the torques provided by the robot's joints at the input and were trained with a laser tracker to provide an improved pose in terms of accuracy and precision. The proposed methods were compared with an Extended Kalman Filter (EKF) where both the LM based pose estimation and the joint torques were utilized.

The proposed methods were validated by machining a NAS 979 part, in which the orientation of the cutting tool was fixed, and free form machining in which the orientation of the cutting tool changed continuously during machining. The machining was performed by a KUKA KR240 R2900 ultra robot on aluminum blocks and the ground truth data was provided by the Leica AT960 laser tracker. As seen from the experimental results for the machining of the NAS 979 part, the EKF, the proposed LSTM network and the SNFIR model were able to reduce the errors on average by 3.89%, 51.76% and 63.74% for position tracking and by 2.71%, 95.32% and 93.98% for orientation tracking, respectively in all three stages of the NAS 979 machining when compared with the pure LM based algorithm. Moreover, the standard deviation of the errors was reduced on average by 4.29%, 36.47% and 52.71% for position tracking and by 3.08%, 93.38% and 90.14% for orientation tracking, respectively. As for free form machining, the EKF, the proposed LSTM network and SNFIR model were able to reduce the errors on average by 5.46%, 54.83% and 82.96% for position tracking and by 2.07%, 26.6% and 72.52% for orientation tracking, respectively in all three stages of the free form machining when compared with the pure LM based algorithm. Furthermore, the standard deviation of errors was reduced on average by 3.86%, 30.54% and 78.75% for position tracking while providing 2.76%, 12.35% and 63.87% better results on average for orientation tracking, respectively. In terms of the achievable absolute position errors, they were 5.47 mm, 2.9 mm and 2.05 mm on average for NAS 979 machining and 5.35 mm, 2.17 mm and 0.86 mm on average for free form machining when using the EKF, the LSTM network and the SNFIR approaches, respectively. As for the achievable absolute orientation errors, they were 0.61°, 0.03° and 0.033° on average for NAS 979 machining and 0.77°, 0.47° and 0.13° on average for free form machining when using the EKF, the LSTM network and the SNFIR approaches, respectively.

As shown, both of the proposed methods significantly increase the accuracy and precision of the standard LM based pose estimation algorithm during robotic machining when the torque information is utilized at the input of the proposed approaches. Moreover, both of the proposed approaches perform significantly better than the EKF algorithm. From the experimental results it is observed that the proposed SNFIR model consistently provides lower position and orientation tracking errors when compared with the proposed LSTM network. This is in addition to the SNFIR model training and running much faster when compared with the LSTM network, all the while providing parsimonious models where only the most important terms accurately representing the data are determined.

In the future, a combination of neural networks and sparse regression with advanced nonlinear models for compensation of camera errors will be developed in order to further improve the absolute accuracy. Moreover, it is planned to extend both of the proposed LSTM and SNFIR methods to be used in a position based visual servoing (PBVS) scheme for improving the machining accuracy of industrial robots.

CRediT authorship contribution statement

Diyar Khalis Bilal: Conceptualization, Methodology, Data Curation, Analysis and interpretation of the data, Software, Validation, Writing – original draft, Writing – review & editing. **Mustafa Unel:** Conceptualization, Methodology, Data Curation, Analysis and interpretation of the data, Software, Validation, Writing – original draft, Writing – review & editing. **Lutfi Taner Tunc:** Funding acquisition, Resources, Investigation, Writing – review & editing. **Bora Gonul:** Data Curation, Software, Investigation.

Declaration of competing interest

One or more of the authors of this paper have disclosed potential or pertinent conflicts of interest, which may include receipt of payment, either direct or indirect, institutional support, or association with an entity in the biomedical field which may be perceived to have potential conflict of interest with this work. For full disclosure statements refer to https://doi.org/10.1016/j.rcim.2021.102262.MUSTAFA UNEL reports financial support was provided by Scientific and Technological Research Council of Turkey (TUBITAK) with grant number 217M078.

Acknowledgment

This work was funded by TUBITAK with grant number 217M078. All authors approved the version of the manuscript to be published.

References

[1] R. Devlieg, Expanding the use of robotics in airframe assembly via accurate robot technology, SAE Int. J. Aerosp. 3 (1846) (2010) 198–203.
[2] A. Klimchik, A. Ambiehl, S. Garnier, B. Furet, A. Pashkevich, Efficiency evaluation of robots in machining applications using industrial performance measure, Robot. Comput.-Integr. Manuf. 48 (2017) 12–29.
[3] K. Hashimoto, A review on vision-based control of robot manipulators, Adv. Robot. 17 (10) (2003) 969–991.
[4] F. Chaumette, Potential problems of stability and convergence in image-based and position-based visual servoing, in: The Confluence of Vision and Control, Springer, 1998, pp. 66–78.
[5] X. Shi, F. Zhang, X. Qu, B. Liu, An online real-time path compensation system for industrial robots based on laser tracker, Int. J. Adv. Robot. Syst. 13 (5) (2016) 1729881416663366.
[6] W. Qu, H. Dong, Y. Ke, Pose accuracy compensation technology in robot-aided aircraft assembly drilling process, Acta Aeronaut. Astronaut. Sin. 32 (10) (2011) 1951–1960.
[7] T. Shu, S. Gharaaty, W. Xie, A. Joubair, I.A. Bonev, Dynamic path tracking of industrial robots with high accuracy using photogrammetry sensor, IEEE/ASME Trans. Mechatronics 23 (3) (2018) 1159–1170.
[8] Comet project, 2013, https://cordis.europa.eu/article/id/90495-nextgeneration-robotic-machining. (Accessed 10 October 2021).
[9] C. Möller, H.C. Schmidt, N.H. Shah, J. Wollnack, Enhanced absolute accuracy of an industrial milling robot using stereo camera system, Proc. Technol. 26 (2016) 389–398.
[10] M. Keshmiri, W.-F. Xie, Image-based visual servoing using an optimized trajectory planning technique, IEEE/ASME Trans. Mechatronics 22 (1) (2016) 359–370.
[11] C. Nissler, B. Stefan, Z.-C. Marton, L. Beckmann, U. Thomasy, Evaluation and improvement of global pose estimation with multiple apriltags for industrial manipulators, in: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation, ETFA, IEEE, 2016, pp. 1–8.
[12] B. Liu, F. Zhang, X. Qu, A method for improving the pose accuracy of a robot manipulator based on multi-sensor combined measurement and data fusion, Sensors 15 (4) (2015) 7933–7952.
[13] K. Claes, H. Bruyninckx, Robot positioning using structured light patterns suitable for self calibration and 3D tracking, in: Proceedings of the 2007 International Conference on Advanced Robotics, Citeseer, Jeju, Korea, 2007.
[14] W.J. Wilson, C.W. Hulls, G.S. Bell, Relative end-effector control using cartesian position based visual servoing, IEEE Trans. Robot. Autom. 12 (5) (1996) 684–696.

[15] F. Janabi-Sharifi, M. Marey, A kalman-filter-based method for pose estimation in visual servoing, IEEE Trans. Robot. 26 (5) (2010) 939–947.
[16] M. Ficocelli, F. Janabi-Sharifi, Adaptive filtering for pose estimation in visual servoing, in: Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, in: Expanding the Societal Role of Robotics in the Next Millennium (Cat. No. 01CH37180), vol. 1, IEEE, 2001, pp. 19–24.
[17] G.E. D'Errico, A la kalman filtering for metrology tool with application to coordinate measuring machines, IEEE Trans. Ind. Electron. 59 (11) (2011) 4377–4382.
[18] G. Alcan, Data Driven Nonlinear Dynamic Models for Predicting Heavy-Duty Diesel Engine Torque and Combustion Emissions (Ph.D. thesis), Sabanci Universiy, 2019.
[19] M.E. Mumcuoglu, G. Alcan, M. Unel, O. Cicek, M. Mutluergil, M. Yilmaz, K. Koprubasi, Driving behavior classification using long short term memory networks, in: 2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive, AEIT AUTOMOTIVE, IEEE, 2019, pp. 1–6.
[20] G. Alcan, E. Yilmaz, M. Unel, V. Aran, M. Yilmaz, C. Gurel, K. Koprubasi, Estimating soot emission in diesel engines using gated recurrent unit networks, IFAC-PapersOnLine 52 (5) (2019) 544–549.
[21] V. Aran, M. Unel, Gaussian process regression feedforward controller for diesel engine airpath, Int. J. Automot. Technol. 19 (4) (2018) 635–642.
[22] G. Zhao, P. Zhang, G. Ma, W. Xiao, System identification of the nonlinear residual errors of an industrial robot using massive measurements, Robot. Comput.-Integr. Manuf. 59 (2019) 104–114.
[23] H. Sun, J. Zhang, R. Mo, X. Zhang, In-process tool condition forecasting based on a deep learning method, Robot. Comput.-Integr. Manuf. 64 (2020) 101924.
[24] P. Aivaliotis, S. Aivaliotis, C. Gkournelos, K. Kokkalis, G. Michalos, S. Makris, Power and force limiting on industrial robots for human–robot collaboration, Robot. Comput.-Integr. Manuf. 59 (2019) 346–360.
[25] W. Zhu, G. Li, H. Dong, Y. Ke, Positioning error compensation on two-dimensional manifold for robotic machining, Robot. Comput.-Integr. Manuf. 59 (2019) 394–405.
[26] C. Hong, J. Yu, J. Zhang, X. Jin, K.-H. Lee, Multimodal face-pose estimation with multitask manifold deep learning, IEEE Trans. Ind. Inf. 15 (7) (2018) 3952–3961.
[27] C. Hong, J. Yu, J. Wan, D. Tao, M. Wang, Multimodal deep autoencoder for human pose recovery, IEEE Trans. Image Process. 24 (12) (2015) 5659–5670.
[28] C. Hong, J. Yu, D. Tao, M. Wang, Image-based three-dimensional human pose recovery by multiview locality-sensitive sparse retrieval, IEEE Trans. Ind. Electron. 62 (6) (2014) 3742–3751.
[29] M. Darcis, W. Swinkels, A.E. Güzel, L. Claesen, Poselab: A levenberg-marquardt based prototyping environment for camera pose estimation, in: 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, CISP-BMEI, IEEE, 2018, pp. 1–6.
[30] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[31] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2003.
[32] Z. Zhang, A flexible new technique for camera calibration, IEEE Trans. Pattern Anal. Mach. Intell. 22 (11) (2000) 1330–1334.
[33] Tensorflow, 2021, https://www.tensorflow.org. (Accessed 10 October 2021).
[34] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937.
[35] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning, Vol. 112, Springer, 2013.
[36] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.
[37] F.J. Romero-Ramirez, R. Muñoz Salinas, R. Medina-Carnicer, Speeded up detection of squared fiducial markers, Image Vis. Comput. 76 (2018) 38–47.
[38] Labview, 2021, https://www.ni.com/en-tr/shop/labview.html. (Accessed 10 October 2021).
[39] Python, 2021, https://www.python.org/. (Accessed 10 October 2021).
[40] NAS979 uniform cutting tests–NAS series metal cutting equipment specifications, 1969, Aerospace Industries Association - National Aerospace Standard.
[41] R.J. Hyndman, G. Athanasopoulos, Forecasting: Principles and Practice, OTexts, 2018.
[42] Extended kalman filter, 2021, https://www.mathworks.com/help/control/ref/extendedkalmanfilter.html. (Accessed 10 October 2021).
[43] Matlab, 2021, https://www.mathworks.com/products/matlab.html. (Accessed 10 October 2021).