The Visual Computer
https://doi.org/10.1007/s00371-024-03296-4

RESEARCH

Multi-camera tracking of mechanically thrown objects for automated in-plant logistics by cognitive robots in Industry 4.0

Nauman Qadeer[1,2] · Jamal Hussain Shah[1] · Muhammad Sharif[1] · Fadl Dahan[3] · Fahad Ahmed Khokhar[4] · Rubina Ghazal[5]

Accepted: 25 January 2024

Abstract
Employing cognitive robots, capable of throwing and catching, is a strategy aimed at expediting the logistics process within Industry 4.0's smart manufacturing plants, specifically for the transportation of small-sized manufacturing parts. Since the flight of mechanically thrown objects is inherently unpredictable, it is crucial for the catching robot to observe the initial trajectory with utmost precision and intelligently forecast the final catching position to ensure accurate real-time grasping. This study utilizes multi-camera tracking to monitor mechanically thrown objects. It involves the creation of a 3D simulation that facilitates controlled mechanical throwing of objects within the internal logistics environment of Industry 4.0. The developed simulation empowers users to define the attributes of the thrown object and capture its trajectory using a simulated pinhole camera, which can be positioned at any desired location and orientation within the in-plant logistics environment of flexible manufacturing systems. The simulation facilitated ample experimentation to be conducted for determining the optimal camera positions for accurately observing the 3D interception positions of a flying object based on its apparent size on the camera's sensor plane. Subsequently, a variety of calibrated multi-camera setups were experimented while placing cameras at identified optimal positions. Based on the obtained results, the most effective multi-camera configuration setup is derived. Finally, a training dataset is prepared for 3000 simulated throwing experiments where the initial part of the trajectory consists of observed interception positions, through derived best multi-camera setup, and the final part consists of actual positions. The encoder–decoder Bi-LSTM deep neural network is proposed and trained on this dataset. The trained model outperformed the current state-of-the-art by accurately predicting the final 3D catching point, achieving a mean average error of 5 mm and a root-mean-square error of 7 mm in 200 real-world test experiments.

Keywords Smart manufacturing systems · Robotic throw-catch approach · Automated in-plant logistics 4.0 · Cognitive robots · Multi-camera simulation · Encoder–decoder Bi-LSTM

1 Introduction

A Flexible Manufacturing System (FMS) is a modern computer-controlled production system having automated material handling and semi-dependent workstations respon-

✉ Fahad Ahmed Khokhar
fahadahmed.khokhar@unifi.it

Nauman Qadeer
nauman.qadeer@fuuast.edu.pk

Jamal Hussain Shah
jhshah@ciitwah.edu.pk

Muhammad Sharif
muhammadsharifmalik@yahoo.com

Fadl Dahan
f.naji@psau.edu.sa

Rubina Ghazal
rubinaghazal@uaar.edu.pk

1 Department of Computer Science, COMSATS University Islamabad, Wah Cantt, Pakistan

2 Department of Computer Science, Federal Urdu University for Arts, Science and Technology, Islamabad, Pakistan

3 Department of Management Information Systems, Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia

4 Department of Mathematics and Information, University of Florence, Florence, Italy

5 Department of Computer Science, University Institute of Information Technology, PMAS Arid Agriculture University, Rawalpindi, Pakistan

Published online: 16 March 2024

sible for efficiently manufacturing parts ranging from low to medium volume [8]. In the manufacturing industry, incorporating computer-controlled machines and robots into the production process offers a multitude of advantages, including improved utilization rates and increased productivity levels [22]. The fast growing trend of using robots in industry is evident by recent world robotics report [19] whose statistics are taken to show the graph in Fig. 1.

The in-plant logistics by throwing and catching robots should be the fastest transportation means due to the direct connection between two workstations. Implementing this throw-catch transportation approach in production systems can result in fully automated processes within flexible manufacturing systems, and it leads to the benefits of optimized utilization of plant facilities, routing flexibility and an increase in productivity volume in addition to fast transportation and all these benefits lead to reduced production cost in FMS [11]. The proposed research aims to investigate this efficient approach.

The trajectory of mechanically thrown object can be observed by cameras. However, the number and placement of these cameras can affect the accuracy of the observations. To reduce observational errors, multiple cameras should be strategically positioned in internal logistics settings to provide a comprehensive view of the trajectory. Such optimum positions need to be determined through rigorous experimentation in such internal logistics vicinity which is usually 3–5 m [10, 31].

The proposed work involves the development of a 3D simulated environment that makes it possible to mechanically throw an object having certain properties and with certain launching conditions. It also allows observation of the resulting captured trajectory through a standard pinhole camera model as such model can effectively describe the perspective projection of most modern cameras [18]. The virtual camera can be positioned anywhere within the 3D simulated environment for internal logistics of the manufacturing plant. The developed simulation is versatile and can be used for any real-world camera by providing its specific characteristics like capturing speed, sensor size, per pixel area, focal length, etc. Figure 2 depicts the whole procedure of projection of 3D simulation world points onto the 2D digital image of the virtual camera.

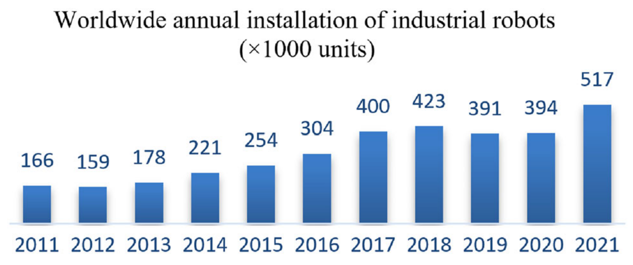


Fig. 1 World robotics report 2022 by Int. Federation of Robotics [19]

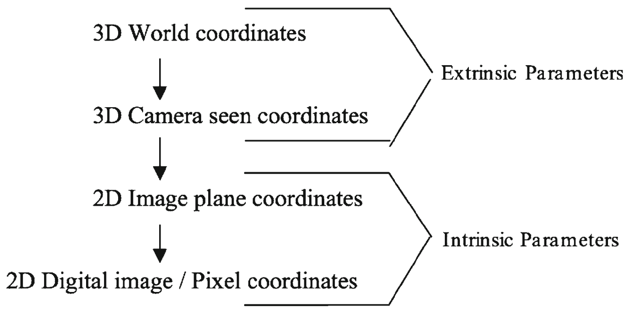

3D World coordinates
3D Camera seen coordinates
2D Image plane coordinates
2D Digital image / Pixel coordinates
Extrinsic Parameters
Intrinsic Parameters

Fig. 2 Process of 2D digital image formation for the virtual camera from the simulated 3D world scene

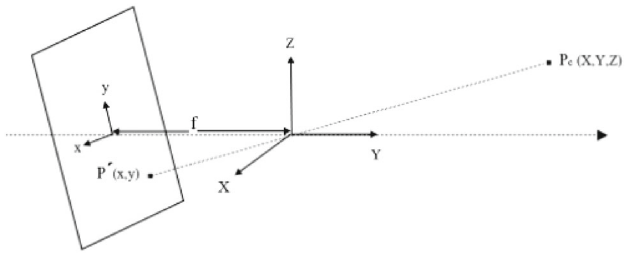


Fig. 3 Used pinhole camera model for 2D image plane mapping of camera seen 3D points

The 3D simulated world points are first converted into 3D camera seen points by taking care of extrinsic parameters (specified location and orientation) of the virtual camera. Then, standard pinhole camera model is used to convert 3D camera seen coordinates into 2D image plane coordinates and finally specified intrinsic parameters of the camera (e.g. sensor size, per pixel area, etc.) are considered for 2D digital image formation in terms of pixels. The standard pinhole camera model, utilized to convert 3D camera seen coordinates into 2D image plane coordinates, is shown in Fig. 3.

Here $P_c(X, Y, Z)$ is the camera seen 3D point that is projected on camera's image plane as 2D point $P'(x, y)$. The two coordinates of P' are derived by following mathematical equations.

$$x = -f \times \frac{X}{Y}. \tag{1.1}$$

$$y = -f \times \frac{Z}{Y}. \tag{1.2}$$

As stated above, any real-world camera can be simulated. However, in simulated experiments of this work, the characteristics of the 'IDS UI-1220RE-M-GL' camera are used. It is because the real-world experiments were carried out using these cameras and using the same camera characteristics enabled comparison of the results of simulated and real-world experiments effectively.

The proposed work can be divided into two parts. The first part is the derivation of the best multi-camera setup, using extensive simulated experiments, for such in-plant logistics settings. This part is further divided into two sub-parts. First, to identify optimal camera positions by extensive experimentation in a simulated environment by throwing a spherical object, with a known radius, and observing its trajectory by placing the virtual camera at multiple locations and observing the errors in reconstructed 3D interception positions. These interception positions are reconstructed from projected 2D positions on the camera's sensor area. Based on the observed errors, the optimal camera positions are identified. Afterward, different combinations of calibrated cameras (placed at the identified optimal positions) are tried in further experimentation to determine the best multi-camera setup with minimal trajectory error.

The second part of this research involves intelligent tracking of thrown objects by a cognitive catching robot. Such a robot observes the initial part of the thrown object's flight trajectory using the best multi-camera setup and predicts the remaining trajectory intelligently with the help of a proposed encoder–decoder Bi-LSTM DNN model. The model was trained on a dataset comprising 3000 trajectories generated through simulated experiments. The evaluation of the proposed model demonstrated its superior performance compared to the current state of the art. It achieved a maximum mean average error (MAE) of 5 mm and root-mean-square error (RMSE) of 7 mm in 200 real-world test experiments, indicating its capability to accurately predict the final catching point.

This manuscript is structured as follows: The next Sect. 2 explores already conducted work related to this research problem. Section 3 explains the proposed methodology. Then, Sect. 4 presents the obtained results. Finally, the paper is concluded in Sect. 5, where planned future work is also discussed.

2 Related work

A lot of work for ball-catching robots found in the literature. For example, the authors of [6] demonstrated their work that was capable of catching two balls simultaneously thrown by humans from 4 to 6 m. The planning algorithm of their work re-plans the robot movement every 20 ms based upon the detected ball through 2 cameras, mounted on the throwing side, and state estimation by UKF (Unscented Kalman Filter). Their reported precision, in both articles [6, 7], was within 2 cm (i.e. 20 mm) for 80 % successful catches. Similarly, automated object detection and tracking have achieved a great accuracy now-a-days as evident by some recent works [2, 9, 21, 30, 33, 34] found in literature.

The robotic throw-catch approach, for material transportation in FMS, was first time proposed in [10]. This approach is feasible for the transportation of only small-sized objects as the aerodynamic instability of large-sized objects is high and this factor makes them highly unpredictable during their flight [12, 14]. Detecting and predicting the interception positions of objects with various shapes, during their flight, is a complex problem that requires a systematic and step-wise solution. In the existing literature [3–5, 10–14, 24–28, 31] on trajectory detection and prediction in in-plant logistics scenarios, researchers opted to utilize a tennis ball as the projectile. It is due to the fact that understanding the aerodynamic properties of an object is critical for accurately predicting its behavior during flight and the aerodynamic properties of a tennis ball are well-studied, and there is a significant body of scientific literature that explores these properties. A review in this context can be found in [23]. Hence, this research work considers only spherical shape objects. However, the proposed work paves the way to extend future research for other shaped objects as well.

The work in [29] used a high-speed 3D video range camera that works under the photonic mixer device (PMD) principle that helps to determine flying object distance with time of flight sensor. Such sensor operates by emitting infrared laser light that hits a target object and then returns to the sensor, where it is detected. A similar effort was made in [3–5] where the single camera, with 87 frames per second (fps) capturing speed, was placed at the catching side. In the initial trajectory of a mechanically thrown tennis ball, its 3D interception positions are determined through photoelectric sensors that were placed at a vertical distance of 40 mm for measuring the initial parameters of the ball. The interception positions of the later part of the trajectory are measured by size-based tracking as interception position determination is more accurate when the flying ball is near to catching side camera. The final 2D impact position accuracy was assessed using a DST touch kit in only 17 test throws. The average error in the final impact position ranged from 1.20 to 3.98 mm. The limitation of this work is that photoelectric sensors are not easily implementable in industry.

The works in [13, 14, 24–28] used mechanical throwing device for throwing tennis ball. After measuring the initial interception positions of the thrown ball through the stereovision of two mounted cameras on the catching side, the remaining trajectory estimation is done by applying different prediction algorithms. For example, in works [26, 27], the fast kNN-based prediction was made for future positions of the ball and an accuracy of 30 mm was achieved as the mean squared error between actual and predicted positions in 92% cases of 150 tested trajectories. In [25], the same authors used a simple one-hidden layer neural network for the remaining trajectory estimation. This neural network was trained by 150 simulated trajectories. During testing, the average prediction error in a simulated environment was approximately 24–26 mm, which is regarded as high for simulated outcomes. In recent studies [15–17, 20], machine learning algorithms have been employed to track mechanically thrown balls for ping-pong-playing robot. The study described in [15–17] constructed a dataset consisting of 614 ball trajectories captured by four ceiling-mounted cameras. A variational auto-encoder deep neural network was trained using 90% of this dataset and subsequently validated using the remaining 10%. The trained model was able to accurately predict the final touch point by analyzing the first 40–50 frames of the ball's flight trajectory, recorded at a speed of 180 frames per second. It had shown an absolute mean error of 40–60 mm when evaluated on 35 test trajectories. This error is high and also suffers from a shortfall of limited testing. The second work, presented in [20], used 3 cameras (right, left, and auxiliary) to observe the flight trajectory of a ping-pong ball thrown through a mechanical device. It trained a dual neural network on 300 trajectories and, upon testing on 30 trajectories; it had shown an absolute mean error of 36.6 mm in final touch point prediction. The limited training and testing was the major drawback of this work. The limitations in existing works for mechanically thrown object tracking are summarized in Table 1.

The limitations of the current state of the art emphasize that tracking of mechanically thrown object's trajectory is still an open research problem due to various factors. There is a need for a large dataset of trajectories in order to have better learning of nonlinear patterns of thrown object trajectories. Furthermore, there is a need to enhance accuracy for measuring interception positions of thrown objects through visual sensors and for this purpose, extensive experimentation is required by placing cameras at different positions. In a real-world environment, producing thousands of trajectories with a multitude of variations in experimental setup is not possible. So a 3D simulated environment is needed in which an object can be thrown with multiple variations and its trajectory can be monitored through simulated cameras following the basic principle of pinhole cameras for perspective projection, as applied in real-world cameras. The work in hand is an effort in this direction.

3 Proposed work

The intention to do this work is to enhance the intelligence of cognitive robots for catching mechanically thrown objects in an internal logistics environment within flexible manufacturing systems of Industry 4.0. The work consists of two parts. The first part is to derive the best multi-camera setup for enhancing the visual tracking of mechanically thrown objects' trajectories. The second part consists of the collection of a dataset, of trajectories captured by derived best

Table 1 Limitations in existing works of mechanically thrown object tracking

Refs	Mechanically thrown obj	Distance range (m)	Spatial position determination Mechanism	Training dataset (Traj.)	Prediction algorithm	MAE	Limited/no training	Limited testing
[3–5, 12]	Tennis ball	2.5−3.0	Using Photoelectric sensors and Size based tracking in single camera's image	–	Extended Kalman Filter	1.20−3.98 mm in 17 test trajectories	✓	✓
[13, 14, 24, 28]	Tennis ball	2.5−3.0	Stereovision by 2 cameras	–	Deterministic Motion model governed by genetic algorithm	5.4 mm in 20 test trajectories	✓	✓
[26, 27]	Tennis ball	2.5−3.0	Stereovision by 2 cameras	2048	KNN	Up to 30 mm in 92% of 150 test trajectories	–	✓
[25]	Tennis ball	2.5−2.7	Simulated Experiments	150	NN (with 1 hidden layer)	24–26 mm in 15 test trajectories	✓	✓
[15–17]	Ping-pong ball	2.7−2.8	Stereovision by 4 cameras	614	Variational auto-encoder deep NN	40–60 mm in 35 test trajectories	✓	✓
[20]	Ping-pong ball	2.7−2.8	Stereovision by 3 cameras	300	Dual NN	36.6 mm in 30 test trajectories	✓	✓

multi-camera setup, and training of the proposed bidirectional LSTM model for intelligent tracking of thrown objects. This paper explains the detailed procedure through involved simulation and algorithms that are used in the derivation of the best multi-camera setup for such internal logistics settings and later on explains the second part of research that utilizes derived multi-camera setup for preparation of trajectories dataset to train proposed intelligent model responsible for accurately predicting final interception position for catching robot. The following two subsections explain each part in detail.

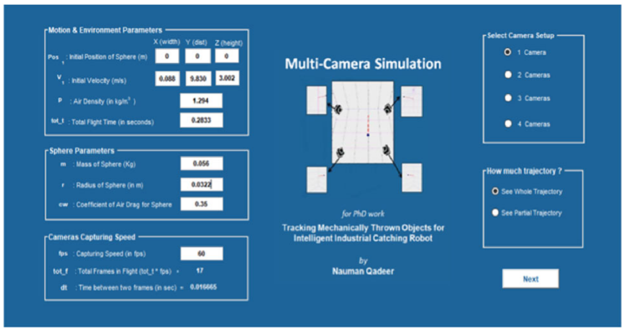
3.1 Proposed best multi-camera setup derivation

For deriving the best multi-camera setup, it is necessary to derive the best camera capturing positions within such an industrial internal logistics environment. It needs the placement of cameras nearly everywhere, within the similar 3D vicinity of internal industrial logistics, and observing obtained reconstructed results about the thrown object's flight trajectory. Since it is nearly impossible in practice, a simulated experimental setup has been developed in accordance with the real-world's camera parameters and experimental setup.

The simulation enables to throwing of spherical objects with any properties. However, the tennis ball was used as a throwing object in simulated and real-world experiments because the same object was used in similar experiments in state of the art and hence it enabled us to compare obtained results accurately with already existing works. Another reason for using a tennis ball is its well-defined properties like a radius of 32.2 mm, mass of 56 g, etc. Additionally, the aerodynamic properties of tennis balls are also well defined like the coefficient of air drag (i.e. cw) as 0.35. The detailed survey about these properties can be found in [1].

The other simulation parameters of the throwing object (like its launching speed, derivation of all three coordinates of its initial velocity from measured launching speed) and camera properties (like its focal length, capturing speed, etc.) are used as per real experiments in order to establish harmony between simulation and real-world results. Moreover, a sample experiment was performed in the real-world and simulated environment by using the same parameters and comparing the 3D interception positions results reconstructed by real and simulated camera-captured frames. A close resemblance in reconstructed results proved the significance of simulation. Interested readers are referred to the previously published paper of this work [32] to know all the details about the sample experiment. Figures 4 and 5 show the first and second screens of the developed simulation that allow setting all above said parameters.

The main screen of the simulation shows that it also allows settings to place multiple (2–4) cameras for capturing


Multi-Camera Simulation

for PhD work
Tracking Mechanically Thrown Objects for Intelligent Industrial Catching Robot
by
Nauman Qadeer

Motion & Environment Parameters

	X (width)	Y (dist)	Z (height)
Pos$_1$: Initial Position of Sphere (m)	0	0	0
V$_1$: Initial Velocity (m/s)	0.088	9.830	3.002
P : Air Density (in kg/m^3)	1.294		
tot_t : Total Flight Time (in seconds)	0.2833		

Sphere Parameters

m : Mass of Sphere (Kg)	0.056
r : Radius of Sphere (in m)	0.0322
cw : Coefficient of Air Drag for Sphere	0.35

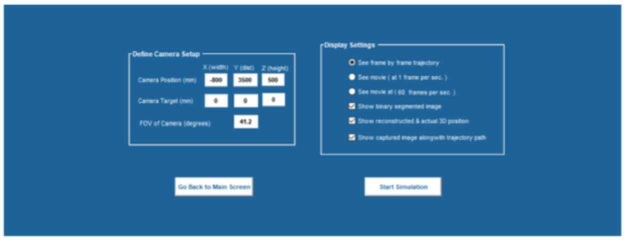
Cameras Capturing Speed

fps : Capturing Speed (in fps)	60
tot_f : Total Frames in Flight (tot_t * fps) =	17
dt : Time between two frames (in sec) =	0.016665

Select Camera Setup: [✓ 1 Camera] 2 Cameras 3 Cameras 4 Cameras

How much trajectory ?: [✓ See Whole Trajectory] See Partial Trajectory

Fig. 4 Developed multi-camera simulation (main screen)


Define Camera Setup

	X (width)	Y (dist)	Z (height)
Camera Position (mm)	-800	3500	500
Camera Target (mm)	0	0	0
FOV of Camera (degrees)	41.2		

Display Settings

[✓ See frame by frame trajectory]
See movie (at 1 frame per sec.)
See movie at (60 frames per sec.)
[✓ Show binary segmented image]
[✓ Show reconstructed & actual 3D position]
[✓ Show captured image alongwith trajectory path]

Fig. 5 Setting camera parameters (screen-2)

mechanically thrown object trajectories. However, at this first stage, only simulated experiments were performed to derive the best camera capturing positions by placing the single camera at multiple positions and analyzing the 3D interception positions accuracy of mechanically thrown objects by analyzing the difference between actual 3D positions of that object and its reconstructed 3D positions that are determined through its size as appeared in its captured trajectory frames. Later on, at the next stage, the combination of multiple cameras will be experimented by placing cameras at derived best positions in both simulated and real-world environments. The algorithm to derive the best camera capturing positions is given as Algorithm 1.

Algorithm 1 starts by capturing trajectories by placing the camera at nearly every reasonably distant position in a simulated 3D world within the reasonable vicinity of the internal logistics environment. The "capture trajectories" process consists of reconstructing trajectories captured by cameras from their videos and then comparing reconstructed 3D object positions with actual positions within its flight trajectory. Hence it determines the accuracy of visually captured trajectories by placing cameras at multiple positions. The step-2 of Algorithm 1 shows that at first, for guessing the best results area, the camera is chosen to be placed at a 500 mm distance (along each dimension). As we must put a limit on axis values, the 3D grid is divided in a way that the camera is considered to be placed from − 500 to 3500 along Y-axis values. At each Y-axis value, the height in camera position (i.e. Z-axis value) is considered from − 500 to 1500 and X-axis positions are considered 1500 mm toward right and left. The camera is always oriented (targeted) toward (0,0,0)

Algorithm 1 Best Camera Capturing Positions Derivation

```
procedure CaptureAndCalculate(Capture trajectories by placing camera at multiple positions)
    Step 1: $C500[\,][\,] \leftarrow \{\}$    ▷ Initialize empty list for holding trajectory results of all 500 mm apart cameras
    Step 2: $C500[\,][\,] \leftarrow$ CaptureTrajectories(500, -1500, 1500, -500, 3500, -500, 1500)
    Step 3: $BCP500[\,] \leftarrow$ CalculateMinimumTrajectoryErrors(C500)
                                      ▷ BCP500 values represent minimum sub-trajectory errors records of C500
    Step 4: $C250[\,][\,] \leftarrow \{\}$    ▷ Initialize empty list for holding trajectories results of all 250 mm apart cameras
                        ▷ Capture trajectories in 250 mm surroundings of minimum sub-trajectory error positions
    Step 5: for $i = 0 \rightarrow 3$ do
            i) $C[\,][\,] \leftarrow$ CaptureTrajectories(250, BCP500[i]][0]-250,C[BCP500[i]][0]+250
                                     C[BCP500[i]][1]-250,C[BCP500[i]][1]+250,
                                     C[BCP500[i]][2]-250,C[BCP500[i]][2]+250)
            ii) Append all records of list $C$ into list C250
        end for
    Step 6: $BCP250[\,] \leftarrow$ CalculateMinimumTrajectoryErrors(C250)
                                      ▷ BCP250 values represent minimum sub-trajectory errors records of C250
    Step 7: $C100[\,][\,] \leftarrow \{\}$    ▷ Initialize empty list for holding trajectories results of all 100 mm apart cameras
                        ▷ Capture trajectories in 100 mm surroundings of minimum sub-trajectory error positions
    Step 8: for $i = 0 \rightarrow 3$ do
            i) $C[\,][\,] \leftarrow$ CaptureTrajectories(100, BCP250[i]][0]-100,C[BCP250[i]][0]+100
                                     C[BCP250[i]][1]-100,C[BCP250[i]][1]+100,
                                     C[BCP250[i]][2]-100,C[BCP250[i]][2]+100)
            ii) Append all records of list $C$ into list C100
        end for
    Step 9: $BCP100[\,] \leftarrow$ CalculateMinimumTrajectoryErrors(C100)
            ▷ Best camera capturing positions are (C100[k][0], C100[k][1], C100[k][2]) where k = BCP100[i] for i=0 to 3
    Output: Best camera positions for capturing in-plant thrown objects trajectory
end procedure
```

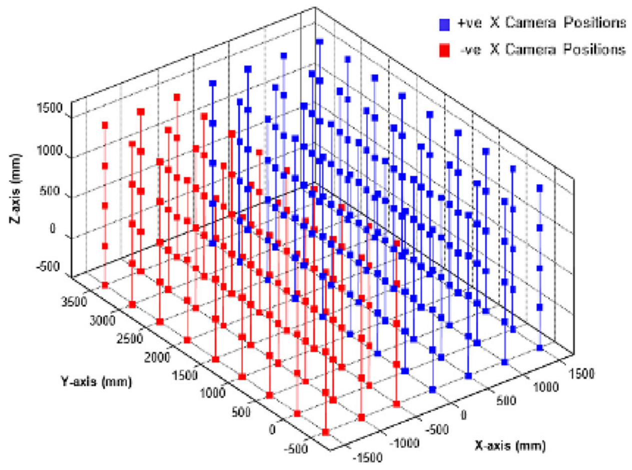


Fig. 6 Initially considered 500 mm apart camera positions

which is the point from where the ball is always launched in a simulated environment. Figure 6 shows 500 mm apart camera positions that are considered for initial experimentation.

Algorithm 2 represents the "capture trajectories" process that consists of capturing trajectories by placing the camera at multiple positions with a particular distance within the specified limits of all three X, Y, and Z axes. In first-time experimentation, the distance is 500 mm and the axes limits are chosen reasonably that represent the 3–5 m vicinity of the internal logistics environment of the industry. However, these limits can be adjusted. For instance, if good results are obtained at the initially chosen value of -500 mm for the Z-axis, then it will show the potential for even better outcomes beyond this value and hence expand the experimentation range by considering the Z-axis value of -1000 mm as well. This iterative process can be further continued.

The obtained results of experimentation are shared in the next section. However, here it can be seen from step-5 of Algorithm 1 that the initial best camera positions obtained, because of minimum trajectory errors, are further experimented with by capturing trajectories at 250 mm apart distance along all three axes surrounding those initially obtained best positions. This process is continued and further refined experimentation of capturing trajectories is done at 100 mm apart distances, along all three axes, surrounding obtained best positions as depicted by step-8 of Algorithm 1.

The process of capturing trajectories, as shown in Algorithm 2, includes determining 3D interception positions of thrown objects by the camera by reconstructing them from the size appearing in captured frames. Afterward, these reconstructed positions (during its flight trajectory) are compared

Algorithm 2 Capture Trajectories and Calculate Errors

```
procedure CaptureTrajectories(Distance, X_start, X_end, Y_start, Y_end, Z_start, Z_end)
    Step 1: $C[\,][\,] \leftarrow \{\}$        ▷ Initialize empty list for holding trajectories results
    Step 2: $cam\_no \leftarrow 0$
        ▷ Determine thrown object's 3D position in 'n' captured frames when camera placed at (i,j,k) position
    Step 3:
    for $i \leftarrow X\_start$ to $X\_end$ do
        for $j \leftarrow Y\_start$ to $Y\_end$ do
            for $k \leftarrow Z\_start$ to $Z\_end$ do
                $C[cam\_no][0] \leftarrow i$
                $C[cam\_no][1] \leftarrow j$
                $C[cam\_no][2] \leftarrow k$
                $n\_t \leftarrow$ Integer$(n/4)$        ▷ 'n' is total no. of frames
                $n\_t1 \leftarrow n\_t + 1$        ▷ last frame no. of Sub-Traj 1
                $n\_t2 \leftarrow (n\_t * 2) + 1$        ▷ last frame no. of Sub-Traj 2
                $n\_t3 \leftarrow (n\_t * 3) + 1$        ▷ last frame no. of Sub-Traj 3
                $n\_t4 \leftarrow n$        ▷ last frame no. of Sub-Traj 4
                $Err\_t1 \leftarrow 0$
                $Err\_t2 \leftarrow 0$
                $Err\_t3 \leftarrow 0$
                $Err\_t4 \leftarrow 0$
                for frame $\leftarrow 1$ to $n\_t1$ do
                    $[X_d, Y_d, Z_d] \leftarrow$ ObjectPositionDetermination(frame, $(i, j, k)$)
                    $Err\_t1 \leftarrow Err\_t1 + \sqrt{(X_d - X_a)^2 + (Y_d - Y_a)^2 + (Z_d - Z_a)^2}$
                end for
                for frame $\leftarrow (n\_t1 + 1)$ to $n\_t2$ do
                    $[X_d, Y_d, Z_d] \leftarrow$ ObjectPositionDetermination(frame, $(i, j, k)$)
                    $Err\_t2 \leftarrow Err\_t2 + \sqrt{(X_d - X_a)^2 + (Y_d - Y_a)^2 + (Z_d - Z_a)^2}$
                end for
                for frame $\leftarrow (n\_t2 + 1)$ to $n\_t3$ do
                    $[X_d, Y_d, Z_d] \leftarrow$ ObjectPositionDetermination(frame, $(i, j, k)$)
                    $Err\_t3 \leftarrow Err\_t3 + \sqrt{(X_d - X_a)^2 + (Y_d - Y_a)^2 + (Z_d - Z_a)^2}$
                end for
                for frame $\leftarrow (n\_t3 + 1)$ to $n\_t4$ do
                    $[X_d, Y_d, Z_d] \leftarrow$ ObjectPositionDetermination(frame, $(i, j, k)$)
                    $Err\_t4 \leftarrow Err\_t4 + \sqrt{(X_d - X_a)^2 + (Y_d - Y_a)^2 + (Z_d - Z_a)^2}$
                end for
                $C[cam\_no][3] \leftarrow Err\_t1$
                $C[cam\_no][4] \leftarrow Err\_t2$
                $C[cam\_no][5] \leftarrow Err\_t3$
                $C[cam\_no][6] \leftarrow Err\_t4$
                $cam\_no \leftarrow cam\_no + 1$
            end for
        end for
    end for
    Output: List $C[nc][7]$
end procedure
```

with actual positions. Hence, it determines the accuracy of visually captured trajectories by placing cameras at multiple positions. Let (X_d, Y_d, Z_d) be the determined 3D position of thrown ball and (X_a, Y_a, Z_a) is its actual position then the magnitude of the difference vector between these two positions can be calculated by the following Eq. 3.1.

$$\text{Magnitude of diff. vector} = \sqrt{(X_d - X_a)^2 + (Y_d - Y_a)^2 + (Z_d - Z_a)^2} \tag{3.1}$$

It shows that visually captured trajectory error can be calculated as the sum of the difference between all corresponding interception positions on two 3D curves of the ball. The first 3D curve consists of the determined ball positions in the reconstructed trajectory scene and the second curve consists of the actual ball positions. So, the whole trajectory error can be calculated by the following Eq. 3.2.

$$\text{Trajectory error} = \sum_{i=1}^{n} \sqrt{(X_{d_i} - X_{a_i})^2 + (Y_{d_i} - Y_{a_i})^2 + (Z_{d_i} - Z_{a_i})^2} \tag{3.2}$$

where ‘n’ is the total number of captured trajectory frames. Depending on the capturing camera position, it is possible that the captured trajectory error could be less in the initial, middle, or final part of the trajectory. So the trajectory is divided into four parts and the corresponding four sub-trajectory errors are calculated separately as depicted by step 3 of Algorithm 2. For example, if the total captured video frames (during the thrown object’s flight) are 17 then these frames can be divided into four segments as shown in Fig. 7. These four sub trajectories (sub1, sub2, sub3 and sub4) actually cover determined errors in first five (1–5), next four (6–9), then next four (10–13) and last 4 four (14–17) frames, respectively, and in this case, these four errors can be calculated by the formulae given in Eqs. 3.3, 3.4, 3.5, and 3.6, respectively.

$$\text{Sub1 traj. Error} = \sum_{i=1}^{5} \sqrt{\left(X_{d_i} - X_{a_i}\right)^2 + \left(Y_{d_i} - Y_{a_i}\right)^2 + \left(Z_{d_i} - Z_{a_i}\right)^2} \tag{3.3}$$

$$\text{Sub2 traj. Error} = \sum_{i=6}^{9} \sqrt{\left(X_{d_i} - X_{a_i}\right)^2 + \left(Y_{d_i} - Y_{a_i}\right)^2 + \left(Z_{d_i} - Z_{a_i}\right)^2} \tag{3.4}$$

$$\text{Sub3 traj. Error} = \sum_{i=10}^{13} \sqrt{\left(X_{d_i} - X_{a_i}\right)^2 + \left(Y_{d_i} - Y_{a_i}\right)^2 + \left(Z_{d_i} - Z_{a_i}\right)^2} \tag{3.5}$$

$$\text{Sub4 traj. Error} = \sum_{i=14}^{17} \sqrt{\left(X_{d_i} - X_{a_i}\right)^2 + \left(Y_{d_i} - Y_{a_i}\right)^2 + \left(Z_{d_i} - Z_{a_i}\right)^2} \tag{3.6}$$

The camera-captured RGB image is converted into HSV colored image first. Then background is subtracted from HSV colored image. After that, noise is removed from a resultant binary image. The ball as a circle is detected from a binary image and its 3D position is reconstructed from the recognized radius of the ball and its 2D center position (in pixels) within the captured frames. Figure 8 below shows some captured images by placing the camera at positions (− 800, 3500, 500) and oriented toward (0,0,0) which is the launching position of the ball.

The same methodology for “object position determination” in camera-captured frame is employed both in simulated and real-world environment. This methodology is explained in detail in following Algorithm 3. Fig. 9 shows

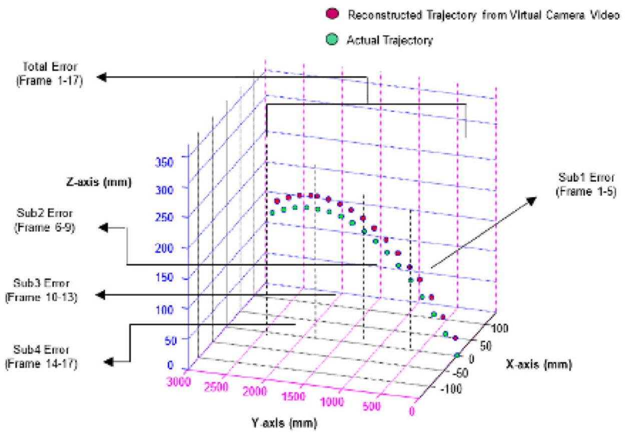
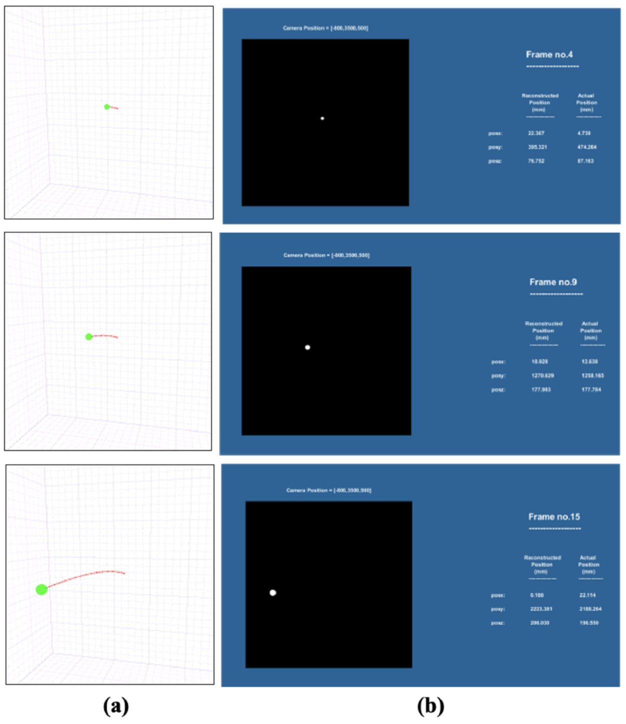


Fig. 7 Trajectory errors when the camera captures trajectory in 17 frames



Camera Position = [-800,3500,500]

Frame no.4

	Reconstructed Position (mm)	Actual Position (mm)
posx:	22.367	4.739
posy:	395.321	474.264
posz:	76.752	87.163

Camera Position = [-800,3500,500]

Frame no.9

	Reconstructed Position (mm)	Actual Position (mm)
posx:	10.928	12.638
posy:	1270.629	1258.165
posz:	177.983	177.784

Camera Position = [-800,3500,500]

Frame no.15

	Reconstructed Position (mm)	Actual Position (mm)
posx:	0.188	22.114
posy:	2223.381	2188.264
posz:	208.030	196.550

Fig. 8 **a** Captured frames 4, 9 & 15 when camera placed at (− 800, 3500, 500) **b** Actual & reconstructed 3D position in corresponding binary segmented image obtained after background subtraction and noise removal

few results from ball detection process in simulated camera captured frame.

Algorithm 2 shows that whenever the process of “capture trajectories” is initiated by placing the camera at multiple (particular distance apart) positions, it returns a list of results in terms of multiple records. Each record in the list represents obtained results for a particularly placed camera and that record has 07 fields. The first three represent the three coor-

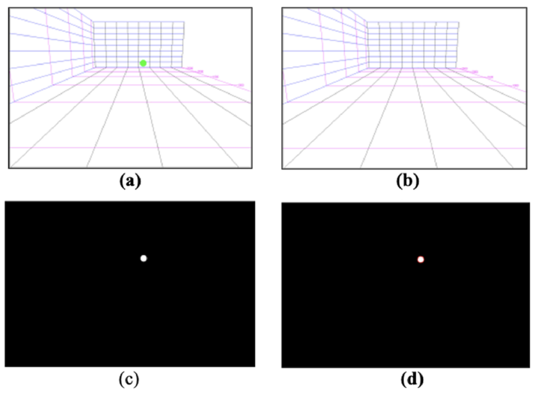


Fig. 9 **a** Simulated camera's captured frame **b** Background image **c** Binary image after background subtraction and noise removal **d** Detected ball through CHT algorithm

dinates for the position of the camera and the last 04 fields represent four sub-trajectory errors obtained while trajectory captured by that camera placement. In order to find the best capturing position against each sub-trajectory error, the four minimum sub-trajectory errors have to be calculated using the Algorithm 4 that returns a list of Best Camera Trajectory (BCTx) that have 04 values pointing toward the indexes of Cx which is in fact the list of all experimented camera positions that are '*x*' distance apart. In this way, the BCTx points toward four camera positions where best-captured trajectories were found due to corresponding minimum four sub-trajectory errors.

So recalling Algorithm 1, the camera is placed on all possible 500 mm apart positions, and then their minimum trajectory errors are calculated to derive the best positions. Then, in phase II, the camera is placed in all possible 250 mm apart positions (adjacent to best-derived positions in the earlier phase). Again, their minimum trajectory errors are calculated and further derive the best positions. Finally, the same process is repeated with 100 mm apart positions surrounding derived positions of phase II and hence final best camera capturing positions are derived in phase III. All these experimentation results are shown in Sect. 4. In the subsequent section, the results are presented for both simulated and real-world environments. Multiple cameras are strategically positioned at the derived optimal locations, enabling the determination of 3D interception positions through stereovision among these cameras. So this experimentation provides the best multi-camera network resulting in the most accurate observation of the trajectory of mechanically thrown objects within the vicinity of the internal logistic environment of Industry 4.0.

Algorithm 3 Object Position Determination

```
procedure ObjectPositionDetermination(Frame, $\text{CamPos}_x$, $\text{CamPos}_y$, $\text{CamPos}_z$)
    Input: A frame (i.e. captured image) and Camera Position ($\text{CamPos}_x$, $\text{CamPos}_y$, $\text{CamPos}_z$)
    Output: Determined 3D interception position $(X_d, Y_d, Z_d)$ of thrown object (i.e. ball) in frame
    Step 1: The input RGB colored image is converted into HSV colored image.
    Step 2: Background is subtracted from HSV colored image by using bitwise XOR operation.
    Step 3: The median filter is applied, on resultant binary image, in order to remove noise.
    Step 4: Then Circular Hough Transform (CHT) algorithm is applied to detect ball in image. The CHT algorithm returns the detected circle's center in terms of x and y pixel and its radius (i.e. $r_{pix}$ )) that is also in terms of pixels..
    Step 5: Calculate the values for $travelled\_pixel_x$ and $travelled\_pixel_y$ by getting the difference between detected ball center pixel values and first frame's corresponding pixel values.
    Step 6: The distance of ball from camera (i.e. d) is determined by the formula of given Equation.
        $d \leftarrow \frac{f \times r}{r_{\text{pix}} \times \text{pixel\_pitch}}$
    here $r$ is the actual radius of the ball, which is 32.2 mm, $f$ is the focal length, and pixel_pitch is per pixel area on the camera's sensor. Their values are 6 mm and 0.006 mm, respectively, which are exactly the same values as the used real camera, i.e., IDS UI-1220RE-M-GL.
    Step 7: Determine the y coordinate of the ball position.
        $Y_c \leftarrow \begin{cases} \text{CamPos}_Y - d & \text{if camera is placed at positive Y-axis} \\ \text{CamPos}_Y + d & \text{if camera is placed at negative Y-axis} \end{cases}$
    Step 8: Determine X and Z coordinates of the ball position.
        $X_c \leftarrow \frac{d \times \text{travelled\_pixel}_x \times \text{pixel\_pitch}}{f}$
        $Z_c \leftarrow \frac{d \times \text{travelled\_pixel}_y \times \text{pixel\_pitch}}{f}$
    Step 9: Actual 3D interception position of ball (i.e. $(X_w, Y_w, Z_w)$ is derived through following transformations of $(X_c, Y_c, Z_c)$ with rotation and translation matrices of camera.
        $\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \leftarrow \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}^{-1} \times \begin{bmatrix} X_c + T_x \\ Y_c + T_y \\ Z_c + T_z \end{bmatrix}$
end procedure
```

3.2 Proposed intelligent tracking for cognitive catching robot

For the applicability of automated in-plant logistics by robotic throw catch approach for smart manufacturing in Industry 4.0, the mechanically thrown object should be accu-

Algorithm 4 Calculate Minimum Trajectory Errors

```
procedure CalculateMinimumTrajectoryErrors($Cx[nc][7]$)
    Input: List $Cx[nc][7]$ containing captured trajectories
    $Cx$ contains captured trajectories results for $nc$ experimented camera positions where all those positions are at $x$ mm apart. It's each record contains 7 fields where the first 3 represent camera's position and last 4 represents obtained sub-trajectory errors
    Output: $BCTx[4]$ containing indexes of $Cx$ for best captured trajectories
    Step 1: $BCTx[4] \leftarrow \{\}$    ▷ Initialize output list of indexes as an empty list
    Step 2: $i \leftarrow 3$
    ▷ Each record of $Cx$ has column indexes 3 to 6 for observed errors of sub-traj-1 to sub-traj-4 respectively
    Step 3:
    for $j \leftarrow 0$ to 3 do
        $min \leftarrow Cx[0][j]$    ▷ Assume first trajectory has minimum sub-trajectory error
        $min\_loc \leftarrow 0$    ▷ Compare assumed minimum sub-trajectory error with all remaining trajectories
        for $k \leftarrow 1$ to $nc - 1$ do
            if $Cx[k][i] < min$ then
                $min \leftarrow Cx[k][i]$
                $min\_loc \leftarrow k$
            end if
        end for
        $BCTx[j] \leftarrow min\_loc$
        $i \leftarrow i + 1$
    end for
end procedure
```

rately caught by intelligent cognitive robots. The derived best multi-camera network, for such industrial settings, helps it to accurately observe 3D spatial coordinates of the thrown object in its initial flight trajectory. But such robots also need to be intelligently trained so that they can accurately predict the remaining trajectory, in real time, by observing the initial flight trajectory of the thrown object. The mechanically thrown object's trajectory tracking and prediction is still an open research problem because of several factors among which the major factor is the nonlinear nature of the thrown object's trajectory.

Consequently, a large dataset of trajectories is required to enhance the learning of nonlinear patterns in mechanically thrown object trajectories. However, generating thousands of trajectories with numerous variations in the experimental setup is not feasible in a real-world environment. Therefore, the created 3D simulated environment enables the throwing of objects with multiple variations, while their trajectories can be tracked using simulated cameras based on the fundamental principles of the pinhole camera model and perspective projection (similar to real-world experimental cameras). The simulation not only allows for camera placement anywhere within the 3D space of the internal logistics environment but also facilitates trajectory capture at any desired capturing speed (frames per second). Thus, it fulfills the requirement for extensive throwing experimentation to gather trajectory data essential for training supervised learning algorithms, particularly deep recurrent neural networks such as RNN and LSTM.

Using the optimal camera setup in a simulated environment, a total of 3000 experiments were performed by minor adjustments in launching parameters (± 0.001 m/s in launching speed as well as ± 0.001 in azimuth and inclination angles) of the thrown ball. Two trajectories are recorded, as two time series, against each experiment. The first time series comprises the perceived interception positions as observed by the cameras, while the second time series contains the actual 3D interception positions. A training dataset needs to be prepared for the throws, where each dataset entry consists of two parts. The initial portion of the trajectory comprises 2/3 of the positions perceived by the observing cameras, while the remaining 1/3 represents the actual positions of the thrown ball.

The proposed deep neural network model, called encoder–decoder Bi-LSTM, is introduced that is capable of accurately predicting the last 1/3 part of the thrown object's trajectory by utilizing the initial 2/3 portion as observed by the cameras. This model has demonstrated excellent performance in identifying patterns in such many-to-many ($m \times n$) mapping problems involving nonlinear time series data. The proposed model architecture is depicted in Fig. 10.
This architectural representation illustrates a proposed intelligent tracking model designed to map a sequence of m input 3D positions to 'n' output 3D positions. The model's input shape is defined as (m, 3), indicating that encoder Bi-LSTM gets initial 'm' three-dimensional interception positions of observed trajectory as input. In fact, the encoder Bi-LSTM consists of two LSTMs (forward and backward) with 50 neurons each.

The output of the encoder Bi-LSTM is passed through a 'Relu' layer and then fed repeatedly n times to the decoder Bi-LSTM using the RepeatVector operation. The decoder Bi-LSTM also consists of two LSTMs (forward and backward) with 50 neurons each. The output of the decoder is processed through a 'Relu' layer and a time-distributed dense layer to represent 3 features (X, Y, and Z position coordinates) at each time step.

The final output of the model is represented as $n \times 3$, representing the last 'n' 3D predicted positions of the thrown object. The decoder Bi-LSTM incorporates the current input, RepeatVector value, and the previous output's hidden state. The RepeatVector operation is used exclusively for repeating the output of encoder LSTM and it does not possess any trainable parameters. Both encoder and decoder Bi-LSTM have 100 input neurons (i.e. 50 neurons for each forward and backward LSTM). The proposed tracking model is evaluated on multiple testing throws and it exhibited outstanding accu-

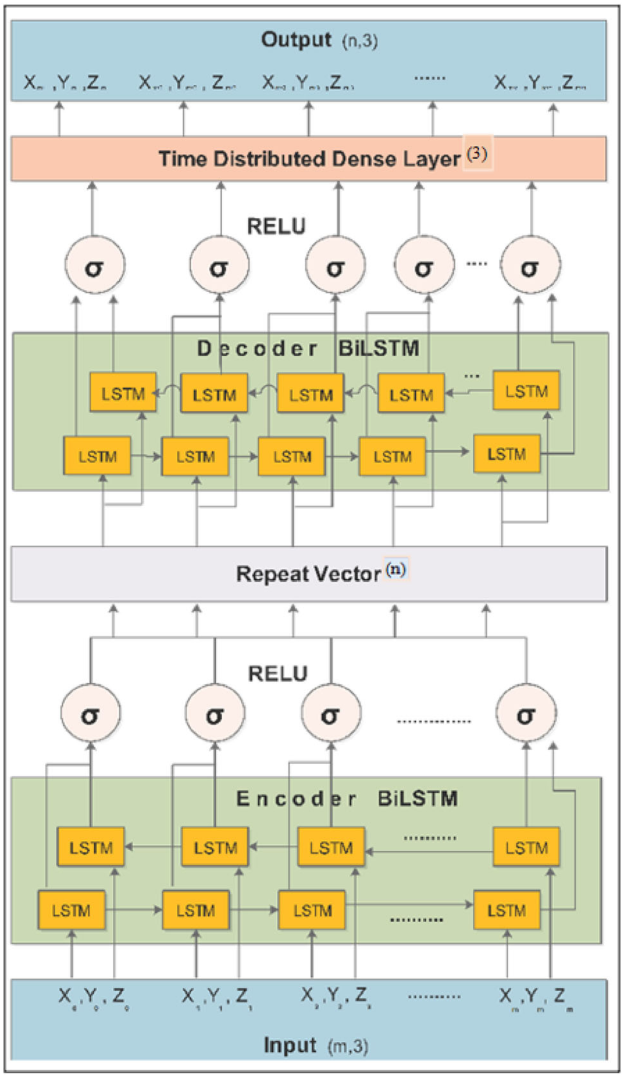


Fig. 10 Proposed intelligent tracking model architecture

racy. The detailed results of these evaluations are presented in the next section.

4 Results and discussion

During real-world experiments, a specially designed mechanical device is utilized to throw the tennis ball. This device incorporates a spring mechanism, and the ball is thrown by harnessing the kinetic energy generated through the stretching or compressing of the spring. The process of launching the tennis ball is visually depicted in Fig. 11.

The thrown ball's launching speed is accurately determined using a radar gun. The initial velocity is then derived based on this measured speed and the throwing angles. Readers interested in a comprehensive explanation of this derivation are referred to the previously published paper of this work [29], which provides detailed insights into the real and simulated experimental setups. The approximate distance separating the throwing point from the final catching point is 3 m. All simulation parameters, including the launching parameters of the thrown object and camera properties, are replicated from the real-world experimental setup. Specifically, the IDS imaging system's camera model "UI-1220RE-M-GL" is employed in real-world experiments, and the characteristics of this camera are summarized in Table 2.

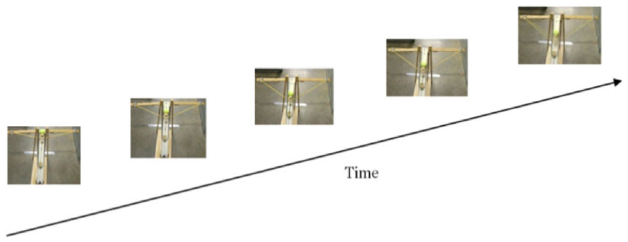


Fig. 11 Mechanical ball launching in real world experiments

Table 2 Characteristics of used camera "IDS UI-1220RE-M-GL"

Feature	Value
Sensor area	$4.512 \times 2.880\,\text{mm}^2$
Resolution	752×480
Pixel pitch (per pixel distance)	$6\,\mu\text{m}$
Focal length	$\backsim 6\,\text{mm}$
Experimented FOV in degrees	41.2°
Experimented capturing speed	60 fps

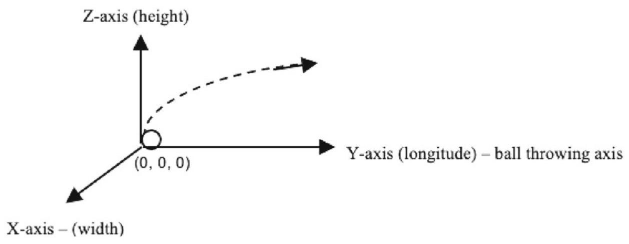


Fig. 12 Considered X, Y, and Z-axis in experimentation

The imaging process in simulated cameras used a standard pinhole camera model that employs perspective projection, as the case in real-world cameras. It means, that viewed objects are projected smaller when distant from the camera and projected larger when near to the camera. In the simulation, the ball is initially launched from point (0, 0, 0) and the axes of its movement are illustrated in Fig. 12.

As explained before, the proposed work consists of two parts. The first part is to derive the best multi-camera setup for enhancing the visual tracking of mechanically thrown objects' trajectories. Then the second part consists of the preparation of the trajectories dataset, using a derived multi-camera setup, and then the training and testing of

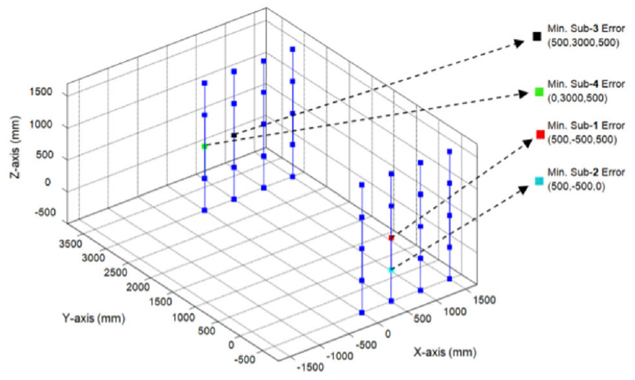


Fig. 13 Minimum observed four sub-trajectory error's positions in initial 500 mm apart experimentation

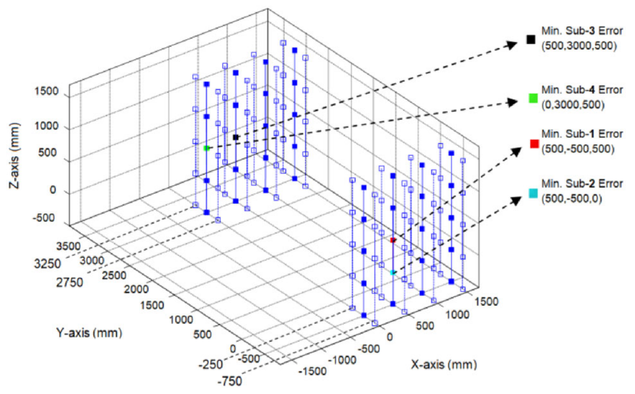


Fig. 14 Camera placement at all 250 mm apart positions surrounding initially best identified positions

the proposed Bi-LSTM model for intelligent tracking of thrown objects. The methodologies employed in both parts are explained, in detail, in the previous section. However, the experimentation results obtained in each part are given in the following two subsections.

4.1 Results in deriving best multi-camera setup

Recalling Algorithm 1, the simulated experimentation initiates by systematically positioning the camera at regular intervals of 500 mm within the designated ranges of the X, Y, and Z axes and then their four minimum sub-trajectory errors are calculated to derive the best camera positions at the first place. In simulated experiments, the initial velocities (along all three axes of the launched tennis ball) are tried to keep the same as their measured values in conducted real-world experiments. When the simulated camera is placed at each of the experimented positions, it is oriented toward (0,0,0), i.e. the launching position of the ball. The camera capturing speed is set at 60 frames per second (fps), which is consistent with the speed used in real-world experiments. Using this rate, the trajectory of the ball is recorded in 17 frames. The camera positions, where minimum sub-trajectory errors are observed, are shown in Fig. 13.

The mentioned four sub-trajectory errors specify the minimum observed errors in the camera-captured first five (1–5), the next four (6–9), then the next four (10–13), and the last 4 four (14–17) frames, respectively. As explained previously, these errors are calculated by formulae given in Eq. 3.3 to 3.6 respectively. In the next step, the cameras are placed at a 250 mm distance surrounding these identified positions. Figure 14 shows the further tested 250 mm apart camera positions.

The results obtained, after experimenting on all the above-mentioned positions, show that minimum trajectory errors are still at the same positions. However, the trajectory errors are less at a distance (i.e. Y-axis value) of − 750 mm as

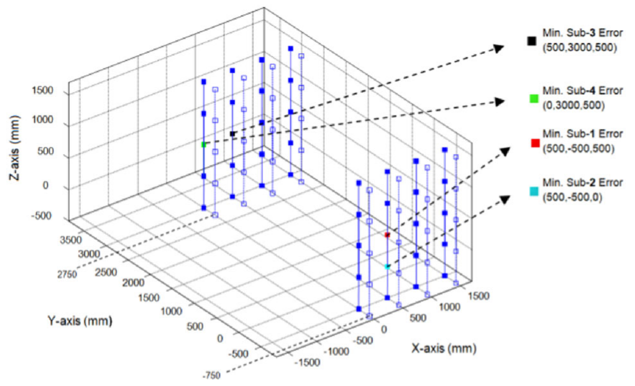


Fig. 15 Better resultant 250 mm apart positions

compared to a Y-axis value of − 250 mm, and also observed that trajectory errors are less at a distance of 2750 mm than at 3250 mm. Figure 15 illustrates that the trajectory of the thrown object is most effectively captured when observed from the throwing side within the distance range of − 750 to − 500 mm along the Y-axis. Furthermore, the results obtained at − 500 mm exhibit superior performance compared to those at − 750 mm.

Fig. 15 also demonstrates that the trajectory of the thrown object is optimally captured when observed from the catching side within the distance range of 2750–3000 mm along the Y-axis. Notably, the results obtained at the Y-axis value of 3000 mm outperform those obtained at 2750 mm. The obtained experimental results have further shown that the errors are less within the Z-axis values between 0 to 500 mm and also X-axis values between 0 to 500 mm. It helped to figure out initially guessed best resultant areas. Those areas are shown in Fig. 16.

As the results on distance − 500 are better than − 750 as well as the results on distance 3000 are better than 2750, so for final refined experiments the selected distances are − 500 and − 600 (near to − 500) and similarly on the catching side, the selected distances are 3000 and 2900 (near to 3000). The

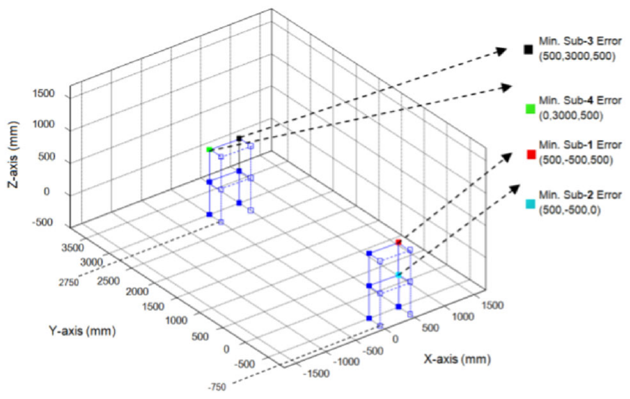


Fig. 16 Initially guessed best resultant areas

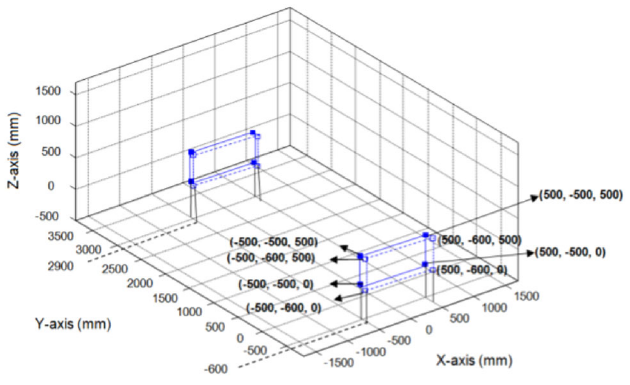


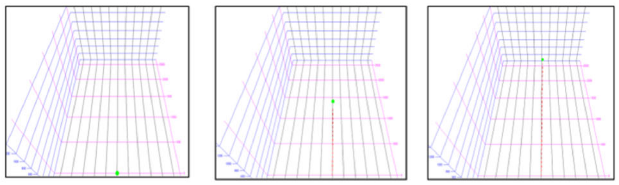
Fig. 17 Finally guessed best resultant areas for refined experiments

camera positions within these areas are considered for refined experiments and the finally selected best resultant areas for refined experimentation are shown in Fig. 17.

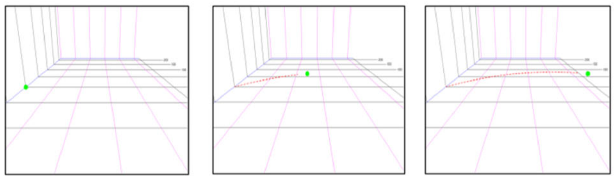
The best results obtained within finally guessed areas make sense because the size of the ball changes significantly within these areas and the reconstruction of the 3D trajectory of a thrown ball, from camera-captured video, is done through its recognized radius in captured video frames. In the following Fig. 18a, b, c, d and e, the initial, middle, and last frames of the trajectory of the thrown ball are shown while the trajectory is captured from the top side, right and left sides, and throwing and catching sides, respectively. Significant variations, in the size of the thrown ball, can be observed when capturing its trajectory from the throwing or catching side, in contrast to top or side views.

Finally, the refined experimentation is done within finally selected areas. This time, cameras are placed at 100 mm apart positions within those areas. Figure 19 shows the camera positions used for refined experiments. Based upon this refined experimentation, Table 3 shows the best-derived camera positions resulting in minimum sub-trajectory errors.

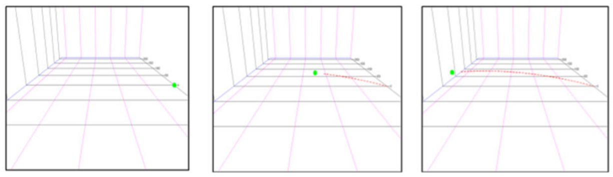
Then further experimentation is employed by different combinations of these best-derived positioned cameras by applying stereovision among them. While trying their combinations, it was ensured that both throwing and catching side cameras must be included for good results. This experimentation is conducted in both simulated and real-world



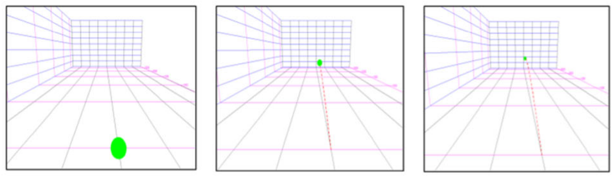
(a) First, middle and last frames of thrown ball trajectory when captured from top side (along with previous trajectory trace).



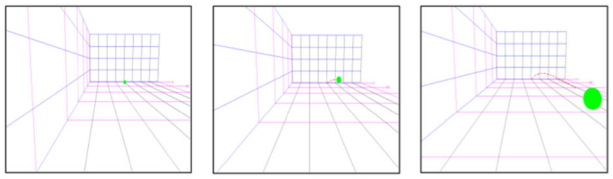
(b) First, middle, and last frames of thrown ball trajectory when captured from one side (along with previous trajectory trace).



(c) First, middle, and last frames of thrown ball trajectory when captured from other side (along with previous trajectory trace).



(d) First, middle, and last frames of thrown ball trajectory when captured from throw side (along with previous trajectory trace).



(e) First, middle, and last frames of thrown ball trajectory when captured from catch side (along with previous trajectory trace).

Fig. 18 First, middle and last frames of thrown ball trajectory

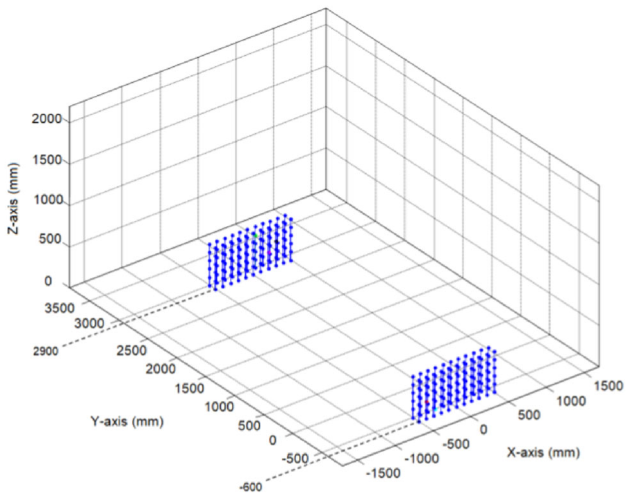
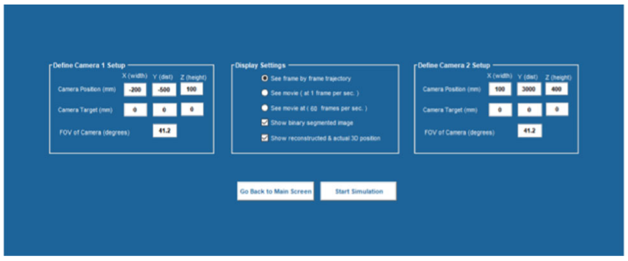


Fig. 19 Camera placement positions considered for refined experiments within the best resultant area

Table 3 Best camera positions based upon min. sub-trajectory error

Camera	Position	Min trajectory error (frames)
C1	– 200, – 500, 100	Frames 1–5
C2	– 400, – 600, 200	Frames 6–9
C3	200, 2900, 200	Frames 10–13
C4	100, 3000, 400	Frames 14–17



Define Camera 1 Setup

	X (width)	Y (dist)	Z (height)
Camera Position (mm)	-200	-500	100
Camera Target (mm)	0	0	0
FOV of Camera (degrees)		41.2	

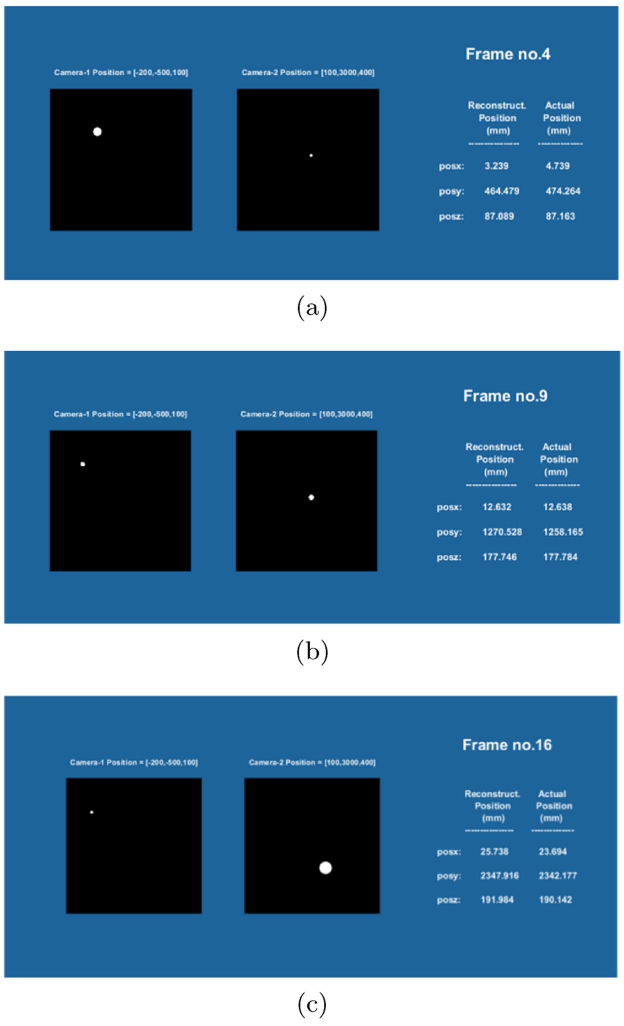
Display Settings

- ● See frame by frame trajectory
- ○ See movie (at 1 frame per sec.)
- ○ See movie at (60 frames per sec.)
- ☑ Show binary segmented image
- ☑ Show reconstructed & actual 3D position

Define Camera 2 Setup

	X (width)	Y (dist)	Z (height)
Camera Position (mm)	100	3000	400
Camera Target (mm)	0	0	0
FOV of Camera (degrees)		41.2	

Fig. 20 Setting parameters for capturing trajectory through C1 and C4



Frame no.4

Camera-1 Position = [-200,-500,100]   Camera-2 Position = [100,3000,400]

	Reconstruct. Position (mm)	Actual Position (mm)
posx:	3.239	4.739
posy:	464.479	474.264
posz:	87.089	87.163

(a)

Frame no.9

Camera-1 Position = [-200,-500,100]   Camera-2 Position = [100,3000,400]

	Reconstruct. Position (mm)	Actual Position (mm)
posx:	12.632	12.638
posy:	1270.528	1258.165
posz:	177.746	177.784

(b)

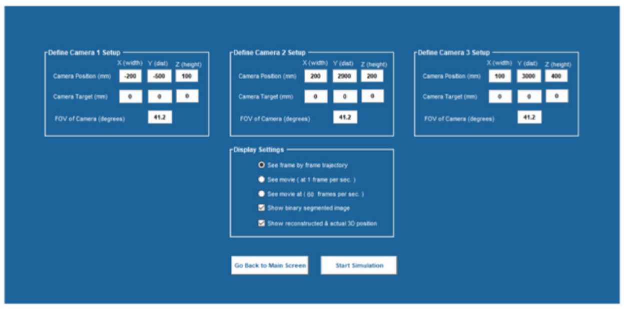
Frame no.16

Camera-1 Position = [-200,-500,100]   Camera-2 Position = [100,3000,400]

	Reconstruct. Position (mm)	Actual Position (mm)
posx:	25.738	23.694
posy:	2347.916	2342.177
posz:	191.984	190.142

(c)

Fig. 21 Actual & reconstructed 3D position in corresponding binary segmented images (**a**. Frame 4 **b**. Frame 9 **c**. Frame 16) obtained after background subtraction & noise removal when trajectory captured through C1 & C4



Define Camera 1 Setup

	X (width)	Y (dist)	Z (height)
Camera Position (mm)	-200	-500	100
Camera Target (mm)	0	0	0
FOV of Camera (degrees)		41.2	

Define Camera 2 Setup

	X (width)	Y (dist)	Z (height)
Camera Position (mm)	200	2900	200
Camera Target (mm)	0	0	0
FOV of Camera (degrees)		41.2	

Define Camera 3 Setup

	X (width)	Y (dist)	Z (height)
Camera Position (mm)	100	3000	400
Camera Target (mm)	0	0	0
FOV of Camera (degrees)		41.2	

Display Settings

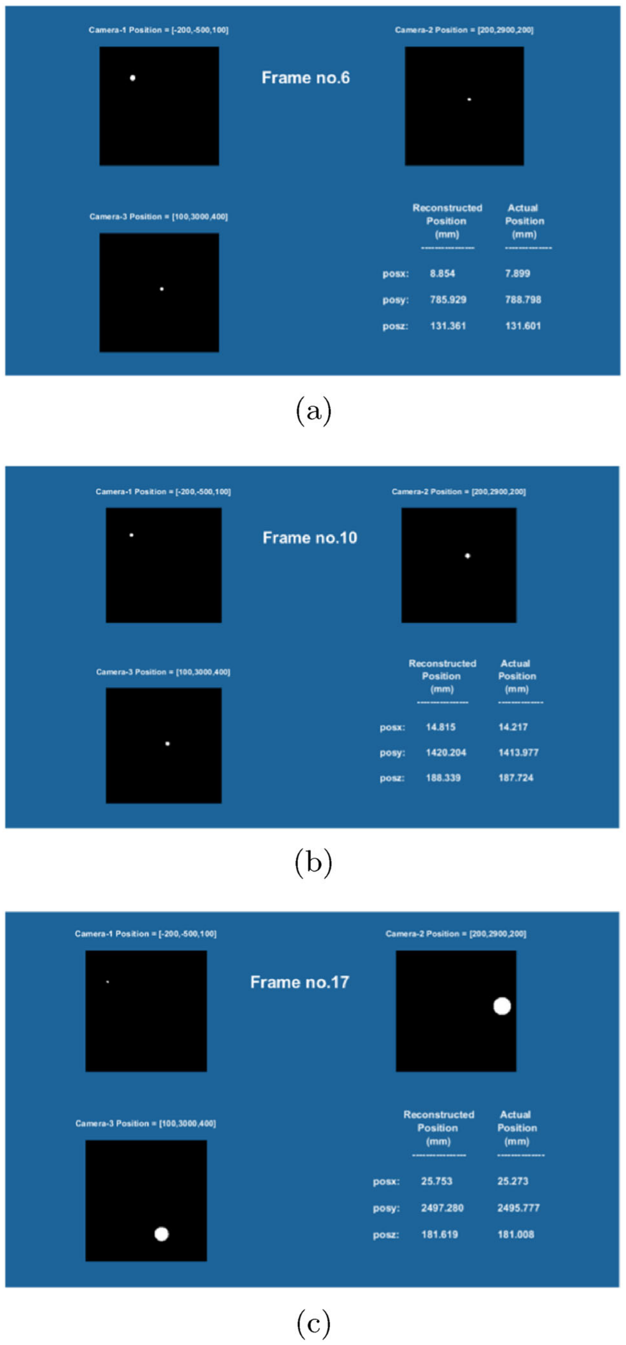
- ● See frame by frame trajectory
- ○ See movie (at 1 frame per sec.)
- ○ See movie at (60 frames per sec.)
- ☑ Show binary segmented image
- ☑ Show reconstructed & actual 3D position

Fig. 22 Setting parameters for capturing trajectory through C1, C3 and C4

environments to determine the optimal multi-camera setup for accurately observing the trajectory of a thrown object in such industrial settings. Figures 20 and 21 show some simulation screenshots when the trajectory was captured through C1 and C4.

Similarly, the following Figs. 22 and 23 show some simulation screenshots when the trajectory is captured through three cameras (C1, C3, and C4). It can be seen that accuracy is further improved when this camera combination is tried.

The 50 simulated experiments were performed with minute variations in the initial launching velocities of the thrown ball. All possible combinations of 2–4 cameras were experimented with. The total trajectory error (calculated by Eq. 3.2) was monitored. Finally, the average error is calculated and presented in Table 4. Table 4 shows that good results are obtained with particular three and four-calibrated camera setups. However, it can be implied from the obtained minimum error result that three calibrated cameras are enough for the best observation of flight trajectory. Moreover, the com-


Camera-1 Position = [-200,-500,100]
Camera-2 Position = [200,2900,200]
Camera-3 Position = [100,3000,400]
Frame no.6
Frame no.10
Frame no.17
(a)
(b)
(c)

(a) Frame no.6

	Reconstructed Position (mm)	Actual Position (mm)
posx:	8.854	7.899
posy:	785.929	788.798
posz:	131.361	131.601

(b) Frame no.10

	Reconstructed Position (mm)	Actual Position (mm)
posx:	14.815	14.217
posy:	1420.204	1413.977
posz:	188.339	187.724

(c) Frame no.17

	Reconstructed Position (mm)	Actual Position (mm)
posx:	25.753	25.273
posy:	2497.280	2495.777
posz:	181.619	181.008

Fig. 23 Actual and reconstructed 3D position in corresponding binary segmented images (**a**. Frame 6 **b**. Frame 10 **c**. Frame 17) obtained after background subtraction and noise removal when trajectory captured through C1, C3 and C4

bination of three cameras (C1, C3, and C4) reconstructed a complete trajectory with minimum error. Hence it was chosen as the optimal multi-camera setup.

The derived optimal multi-camera setup was also tested for 50 real-world tennis ball-throwing experiments. However, the accuracy of the reconstructed final position of the ball could only be tested using a DST touch screen that is placed on the catching side. This screen could measure two coordinates (X-axis and Z-axis values) but its placement (i.e. distance) can measure the Y-axis value as well. The results of these experiments have shown promising accuracy of the final 3D touch point of the ball as the average reconstruction error for both X and Z-axis values was under 2 mm and for Y-axis it was under 4 mm.

4.2 Proposed tracking model results

The proposed model needed a large training dataset of throws and, for this purpose, the 3000 simulation experiments were performed using a derived optimal multi-camera configuration setup. Minor adjustments, in launching parameters, were made for a variety of experiments. Two trajectories are recorded, as two time series, against each experiment. The first time-series comprises the perceived interception positions as observed by the cameras, while the second time series contains the actual 3D interception positions. This approach was adopted to ensure that each training dataset's trajectory should contain initial 3D interception positions as camera-observed positions, while the final 3D interception positions should correspond to the actual positions. This design choice is influenced by the findings of the catching robot's work presented in [12], which highlights the need for a minimum of 80 ms to allow the motor of a contemporary catching robot to accurately position its catching gripper at the predicted final 3D position. So, the final 3D catching position should be predicted at least 80 ms before the flying object reaches its final position.

In the real-world experimentation, the capturing speed of the camera was 60 frames per second. So, using the same capturing speed in simulated experiments, the moving ball was found in 17 frames with a total trajectory time of around 283 ms. To effectively train the proposed intelligent tracking model, it is essential to configure it to receive as input the initial 3D interception positions observed by cameras within the initial 200 ms flight trajectory of the thrown ball. The model should then be capable of predicting the subsequent 83 ms of the flight trajectory. So, each of the 3000 trajectories of the training dataset is composed of 12 interception positions (i.e. 2/3 of trajectory) as observed by cameras and the remaining 5 interception positions (i.e. 1/3 of trajectory) as the last 5 actual positions of the ball in simulation.

The architecture of the proposed intelligent tracking model is already explained in Sect. 3.2. The proposed model of encoder–decoder Bi-LSTM effectively retains time-series information from both forward and backward sequential contexts, making it well-suited for learning patterns in many-to-many time series problems. This model was evaluated using various numbers of tested throws and demonstrated superb accuracy in predicting the last 5 interception positions of a thrown object by observing the first 12 interception

Table 4 Average trajectory error of 50 simulated test experiments(using various multi-camera setups)

Multi-camera setup	Total trajectory error (mm)
$C1 + C3$	65
$C1 + C4$	79
$C2 + C3$	72
$C1 + C2 + C3$	43
$C1 + C2 + C4$	47
$C1 + C3 + C4$	31
$C2 + C3 + C4$	38
$C1 + C2 + C3 + C4$	34

positions, through a derived multi-camera setup to achieve optimal performance. As each predicted interception position has three coordinate values there are a total of 15 calculated error values. The formulae for calculating root-mean-square error (RMSE) and mean absolute error (MAE) are given in the following Eqs. 4.1, 4.2, respectively.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{5}\sum_{j=1}^{3}\left(\text{Predicted}_{ij} - \text{Actual}_{ij}\right)^2}{15}} \tag{4.1}$$

$$\text{MAE} = \frac{\sum_{i=1}^{5}\sum_{j=1}^{3}\left|\text{Predicted}_{ij} - \text{Actual}_{ij}\right|}{15} \tag{4.2}$$

The model is tested gradually from a small to large datasets of simulated experimented throws. Figure 24 depict the prediction error results obtained by testing the proposed model on four distinct test datasets comprising 100, 200, 500, and 1000 simulated throws, respectively. The presented graphs demonstrate consistent error values within the favorable range of 0.5–2.5 mm, indicating the model's effectiveness.

In the following Fig. 25, the comparison between ground truth and predicted values (of X, Y, and Z-axis, respectively) is shown for the last 5 predicted interception positions of a tested thrown ball in the simulated experiment. Within these Figs, the initial 12 values represent the observed interception positions, captured using a multi-camera setup, while the last 5 values correspond to the ground truth and predicted values generated by the proposed intelligent tracking model. The substantial overlap among corresponding predicted and ground truth values demonstrates the high accuracy achieved by the proposed model.

In real-world scenarios, similar experiments were conducted. However, due to the absence of absolute position detector sensors in the experimental setup, it was only possible to measure the final position of the thrown ball using a DST touch screen located on the catching side. The accuracy of predicting the final 3D catching position was assessed by root-mean-square error (RMSE) and mean absolute error

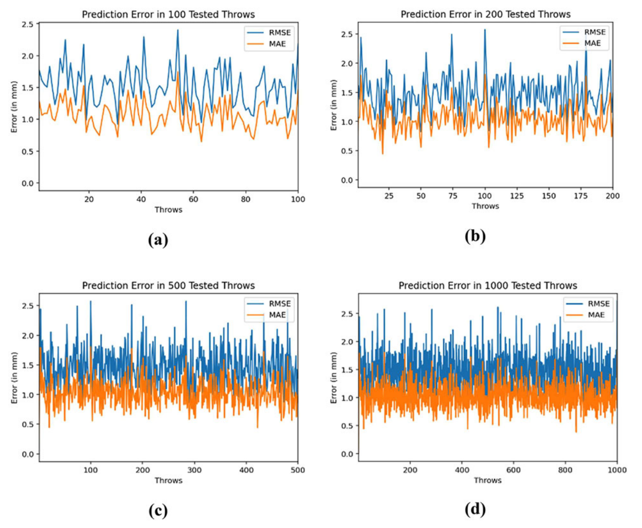
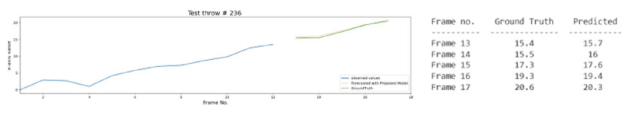
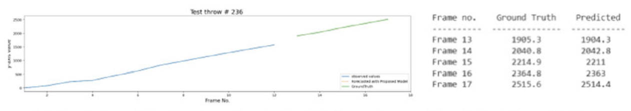


Fig. 24 Prediction error results of proposed model evaluation for different datasets of simulated test throws



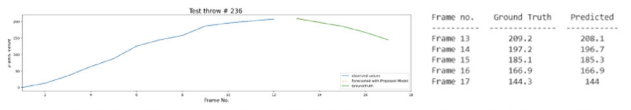
Frame no.	Ground Truth	Predicted
Frame 13	15.4	15.7
Frame 14	15.5	16
Frame 15	17.3	17.6
Frame 16	19.3	19.4
Frame 17	20.6	20.3

(a) Prediction of X-axis values (mm) of last 5 interception positions in test throw # 236



Frame no.	Ground Truth	Predicted
Frame 13	1905.3	1904.3
Frame 14	2040.8	2042.8
Frame 15	2214.9	2211
Frame 16	2364.8	2363
Frame 17	2515.6	2514.4

(b) Prediction of Y-axis values (mm) of last 5 interception positions in test throw # 236



Frame no.	Ground Truth	Predicted
Frame 13	209.2	208.1
Frame 14	197.2	196.7
Frame 15	185.1	185.3
Frame 16	166.9	166.9
Frame 17	144.3	144

(c) Prediction of Z-axis values (mm) of last 5 interception positions in test throw # 236

Fig. 25 Comparison between ground truth and predicted values (of X, Y and Z-axis, respectively) for last 5 predicted positions of a tested throw

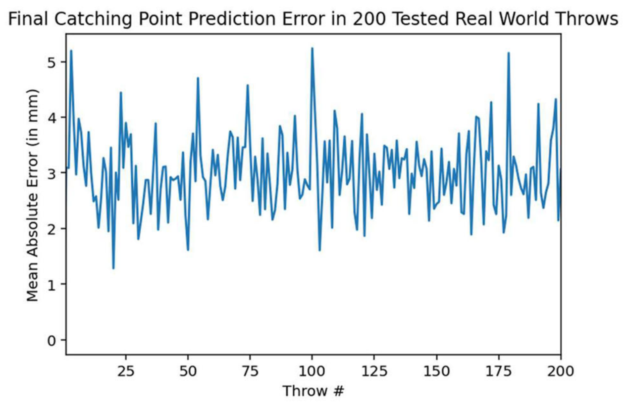
(MAE) that are calculated using the formulae provided in Eqs. 4.3 and 4.4, respectively.

$$\text{RMSE in Final Position Prediction} = \sqrt{\frac{\sum_{i=1}^{3}(Prediction_i - \text{Actual}_i)^2}{3}} \tag{4.3}$$

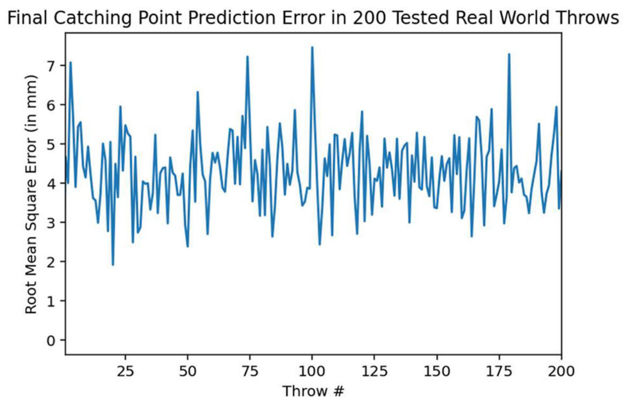
$$\text{MAE Final Position Prediction} = \frac{\sum_{i=1}^{3}\left|Predicted_i - \text{Actual}_i\right|}{3} \tag{4.4}$$

Fig. 26a and b present obtained error results, for final touch point prediction, in terms of MAE and RMSE, respectively. These results are obtained by 200 ball-throwing real-world test experiments.

These obtained results have shown that the proposed intelligent tacking model has shown the ability of final catch-



(a) Final Catching Point Error (MAE)



(b) Final Catching Point Error (RMSE)

Fig. 26 Final touch point prediction error results by proposed model when evaluated on 200 real-world test throws

ing point prediction with a maximum MAE error of 5 mm and RMSE of 7 mm while tested in 200 real-world throwing experiments. The obtained results are compared with some recently implemented works of mechanically thrown object tracking that utilized machine learning techniques for final catching point prediction. This comparative analysis is presented in Table 5. The results demonstrate significant improvement in the accuracy of predicting the final catching point through intelligent tracking by the proposed model.

5 Conclusion and future work

For smart manufacturing in Industry 4.0, the approach of utilizing throwing and catching cognitive robots holds the potential to become the most efficient mode of in-plant logistics in an automated way. However, for the adaptability of this approach in industry, further research is required in two key dimensions. Firstly, the identification of an optimal multi-camera setup is necessary that accurately observe the initial flight trajectory of thrown objects within in-plant industrial settings. Secondly, an advanced machine learning algorithm is needed to enhance the intelligence of the cognitive catching robot. This algorithm should enable the catching robot to accurately predict the final 3D catching position by analyzing the observed initial flight trajectory of mechanically thrown objects.

The proposed work aims to address both of these research dimensions. It involves the creation of a 3D simulation environment that facilitates controlled mechanical throwing of objects within the internal logistics environment of Industry 4.0. Through this simulation, users have the ability to specify the attributes of the thrown object and observe its trajectory by positioning a simulated camera at any desired position and orientation within the 3D environment. To the best of our knowledge, this simulation represents the first attempt in this research domain. The simulation played a crucial role in conducting extensive experimentation to identify optimal camera positions for accurately capturing the 3D interception positions of the flying objects based on their apparent size on the camera's sensor plane. Various calibrated multi-camera setups were tested using the identified optimal positions, and the most effective configuration was determined based on the obtained results.

Subsequently, a training dataset was created, consisting of trajectories of 3000 simulated throwing experiments. The initial part of each trajectory, of the dataset, contained the interception positions observed through the derived optimal multi-camera setup, while the final part consisted of the actual positions. An encoder–decoder bidirectional LSTM neural network model was proposed and trained on this dataset. The performance of the model exceeded the state-of-the-art, achieving a mean average error of 5 mm and a root-mean-

Table 5 Comparison of final catching point prediction accuracy achieved in this work and claimed in other state-of-the-art works for mechanically thrown object tracking

Refs	Year	ML model	Training dataset	Testing dataset	MAE	RMSE
[26]	2016	kNN regression	2048 throws	150 throws	–	30 mm
[28]	2019	Deterministic motion model	1000 throws	20 throws	–	5.4 mm
[15–17]	2020	Variational auto-encoder	614 throws	35 throws	50 mm	–
[20]	2020	Dual neural network	300 throws	30 throws	36.6 mm	–
Proposed	2023	Encoder–decoder Bi-LSTM	3000 throws	200 throws	05 mm	07 mm

square error of 7 mm in predicting the final 3D catching point when tested on 200 real-world experiments.

It should be noted that this work is currently limited to objects with a spherical shape. However, future plans include expanding this research to encompass other regular shapes such as rectangular, square, or cylindrical objects. Additional research is required to track the orientation of such regular-shaped objects during their flight trajectory. The successful implementation of this approach has the potential to transform traditional logistics of small-sized materials in industries and lead to improved efficiency and cost savings.

Acknowledgements The authors would like to thank HEC's National Research Program for Universities of Pakistan to support this research under Project no. 7794.

Author Contributions Conceptualization: N.Q.; Methodology: N.Q.; Validation: N.Q., J.H.S., R.G.; Formal analysis: N.Q., J.H.S., M.S.; Investigation: N.Q.; Data curation: N.Q., J.H.S.; Writing - original draft preparation: N.Q.; Writing - review and editing, N.Q., R.G., F.A.K.; Supervision: J.H.S.; Project administration: J.H.S.; Funding acquisition: J.H.S., F.A.K. and F.D. All authors have read and agreed to the published version of the manuscript.

Funding Open access funding provided by Universitá degli Studi di Firenze within the CRUI-CARE Agreement.

Data Availability The datasets generated during and/or analyzed during the current study are available from the first author (i.e. N.Q.) on reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Alam, F., Tio, W., Watkins, S., et al.: Effects of spin on tennis ball aerodynamics: an experimental and computational study, 324–327 (2007)
2. Ansari, G.J., Shah, J.H., Farias, M.C., et al.: An optimized feature selection technique in diversified natural scene text for classification using genetic algorithm. IEEE Access **9**, 54923–54937 (2021)
3. Barteit, D., Frank, H., Kupzog, F.: Accurate prediction of interception positions for catching thrown objects in production systems. In: 2008 6th IEEE International Conference on Industrial Informatics, pp. 893–898. IEEE (2008)
4. Barteit, D., Frank, H., Pongratz, M., et al.: Measuring the intersection of a thrown object with a vertical plane. In: 2009 7th IEEE International Conference on Industrial Informatics, pp. 680–685. IEEE (2009)
5. Barteit, D.F.: Tracking of thrown objects: catching of mechanically thrown parts for transport in manufacturing. PhD thesis, Vienna University of Technology, Austria (2010)
6. Bauml, B., Wimbock, T., Hirzinger, G.: Kinematically optimal catching a flying ball with a hand-arm-system. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2592–2599. IEEE (2010)
7. Bäuml, B., Birbach, O., Wimböck, T., et al.: Catching flying balls with a mobile humanoid: system overview and design considerations. In: 2011 11th IEEE-RAS International Conference on Humanoid Robots, pp. 513–520. IEEE (2011)
8. Black, J.T., Kohser, R.A.: DeGarmo's Materials and Processes in Manufacturing. John Wiley & Sons, Hoboken (2020)
9. Chen, Z., Wang, J., Sheng, B., et al.: Illumination-invariant video cut-out using octagon sensitive optimization. IEEE Trans. Circuits Syst. Video Technol. **30**(5), 1410–1422 (2019)
10. Frank, H., Wellerdick-Wojtasik, N., Hagebeuker, B., et al.: Throwing objects–a bio-inspired approach for the transportation of parts. In: 2006 IEEE International Conference on Robotics and Biomimetics, pp. 91–96. IEEE (2006)
11. Frank, H., Barteit, D., Kupzog, F.: Throwing or shooting-a new technology for logistic chains within production systems. In: 2008 IEEE International Conference on Technologies for Practical Robot Applications, pp. 62–67. IEEE (2008a)
12. Frank, H., Barteit, D., Meyer, M., et al.: Optimized control methods for capturing flying objects with a cartesian robot. In: 2008 IEEE Conference on Robotics, Automation and Mechatronics, pp. 160–165. IEEE (2008b)
13. Gayanov, R., Mironov, K., Kurennov, D.: Estimating the trajectory of a thrown object from video signal with use of genetic programming. In: 2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), pp. 134–138. IEEE (2017)
14. Gayanov, R., Mironov, K., Mukhametshin, R., et al.: Transportation of small objects by robotic throwing and catching: applying genetic programming for trajectory estimation. IFAC-PapersOnLine **51**(30), 533–537 (2018)
15. Gomez Gonzalez, S.: Real time probabilistic models for robot trajectories. PhD thesis, Technische Universitat Darmstadt (2019)
16. Gomez-Gonzalez, S., Nemmour, Y., Scholkopf, B., et al.: Reliable real-time ball tracking for robot table tennis. Robotics **8**(4), 90 (2019)
17. Gomez-Gonzalez, S., Prokudin, S., Scholkopf, B., et al.: Real time trajectory prediction using deep conditional generative models. IEEE Robot. Autom. Lett. **5**(2), 970–976 (2020)
18. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2003)
19. IFR.: World robotics report by int. federation of robotics. https://ifr.org/ifr-press-releases/news/wr-report-all-time-high-with-half-a-million-robots-installed, accessed: 2023-11-29 (2022)
20. Lin, H.I., Yu, Z., Huang, Y.C.: Ball tracking and trajectory prediction for table-tennis robots. Sensors **20**(2), 333 (2020)
21. Lin, X., Sun, S., Huang, W., et al.: EAPT: efficient attention pyramid transformer for image processing. IEEE Trans. Multimed., 1–13 (2021)
22. Maddikunta, P.K.R., Pham, Q.V., Prabadevi, B., et al.: Industry 5.0: a survey on enabling technologies and potential applications. J. Ind. Inf. Integr. **26**, 100257 (2022)
23. Mehta, R., Alam, F., Subic, A.: Review of tennis ball aerodynamics. Sports Technol. **1**(1), 7–16 (2008)

24. Mironov, K.: Transport by robotic throwing and catching: Accurate stereo tracking of the spherical object. In: 2017 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), pp. 1–6. IEEE (2017)
25. Mironov, K., Pongratz, M.: Applying neural networks for prediction of flying objects trajectory. **17**(6):33–37 (2013)
26. Mironov, K., Pongratz, M.: Fast kNN-based prediction for the trajectory of a thrown body. In: 2016 24th Mediterranean Conference on Control and Automation (MED), pp. 512–517. IEEE (2016)
27. Mironov, K., Vladimirova, I., Pongratz, M.: Processing and forecasting the trajectory of a thrown object measured by the stereo vision system. IFAC-PapersOnLine **48**(11), 28–35 (2015)
28. Mironov, K., Gayanov, R., Kurennov, D.: Observing and forecasting the trajectory of the thrown body with use of genetic programming (2019)
29. Moller, T., Kraft, H., Frey, J., et al.: Robust 3D measurement with PMD sensors. Range Imaging Day Zurich **7**(8), 3 (2005)
30. Nadeem, A., Rizwan, K., Mehmood, A., et al.: A smart city application design for efficiently tracking missing person in large gatherings in Madinah using emerging IoT technologies. In: 2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC), pp. 1–7. IEEE (2021)
31. Pongratz, M., Pollhammer, K., Szep, A.: Koros initiative: automatized throwing and catching for material transportation. In: International Symposium On Leveraging Applications of Formal Methods, Verification and Validation, pp. 136–143. Springer (2011)
32. Qadeer, N., Shah, J.H., Sharif, M., et al.: Intelligent tracking of mechanically thrown objects by industrial catching robot for automated in-plant logistics 4.0. Sensors **22**(6), 2113 (2022)
33. Shen, Y., Lin, X., Gao, Y., et al.: Video composition by optimized 3D mean-value coordinates. Comput. Animat. Virtual Worlds **23**(3–4), 179–190 (2012)
34. Zhu, H., Sheng, B., Lin, X., et al.: Foreground object sensing for saliency detection. In: Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval, pp. 111–118 (2016)

Nauman Qadeer (PhD, Senior Member IEEE) is serving as assistant professor in Computer Science department of Federal Urdu University for Arts, Science & Technology, Islamabad, Pakistan. He is concurrently fulfilling the roles of Director of IT for the Islamabad campus of this university, as well as serving as the Graduate Program Coordinator for the Computer Science department. He has been serving in this university since 2005. Prior to this job, he also served as Lecturer of Computer Science in University of Agriculture Faisalabad (UAF) from 2002 to 2005. He received Ph.D. in Computer Science (Computer Vision) from COMSATS University Islamabad. He also holds M.S. in Computer Science (with distinction) from UAF. He had been awarded gold medal for achieving first position in M.Sc. (Computer Science) from B.Z. University, Multan. He has authored 15+ papers in esteemed journals and international conferences. He also served as invited peer reviewer for 45+ papers of reputable WoS indexed journals. His research interests include computer vision, deep learning, pattern recognition, robotics and automation, multi-agent systems, intelligent video surveillance and smart city applications. He is senior member IEEE as well as an active member of IEEE Consultants Network and several IEEE societies like IEEE Robotics and Automation, IEEE Computational Intelligence Society, EMB BioRobotics, EMB BIIP, IEEE Industrial Electronics and IEEE SIGHT.

Jamal Hussain Shah received the Ph.D. degree in pattern recognition from the University of Science and Technology China, Hefei, China. Since 2008, he has been in the education field. He is currently serving as Tenured Associate Professor at COMSATS University Islamabad, Wah Campus, Pakistan. His areas of specialization are automation and pattern recognition. He has over 60 publications in IF, SCI and ISI journals and in national and international conferences. His research interests include deep learning, algorithms design and analysis, machine learning, image processing, and big data.

Muhammad Sharif (PhD, Senior Member IEEE) is serving as Professor at COMSATS University Islamabad, Wah Campus, Pakistan. He has completed his PhD in Image Processing from COMSATS Institute of IT, Islamabad in 2013. He got his MS (CS) degree from the same institute in 2007 and MCS degree from Quaid-e-Azam University, Islamabad Pakistan in 1995. He has worked one year in Alpha Soft UK based software house as a developer/team leader in 1995. He is an Oracle Certified Professional in Developer Track. He has adopted teaching profession since 1996 to date. He has more than 160 research publications in IF, SCI and ISI journals as well as in national and international conferences and obtained 100+ Impact Factor. He has been supervised 45+ MS (CS) thesis till date. Currently, he is supervising 05 PhD (CS) students and co-supervising 10 PhD (CS) students. Few of his students successfully defended PhD (CS) thesis. In addition, more than 350 undergraduate students have successfully completed their degree project work under his supervision. His research interests include Medical Imaging, Biometrics, Computer Vision, Machine Learning and Agriculture/Plants imaging. He is being awarded with COMSATS Research Productivity Awards continuously since 2010 till now. He has been served in the TPC for the IEEE FIT 2013–17. He is also currently serving as an Associate Editor for IEEE Access Journal and Guest Editor in four Journal special issues as well as reviewer for many well reputed journals. He has headed the Department of Computer Sciences in COMSATS University Islamabad, Wah Cantt. Campus from 2008 to 2011 and successfully achieved the targeted outputs. He also attended a number of prestigious national and international conferences throughout his career.

Fadl Dahan is Associate Professor of Artificial Intelligence at department of Management Information System in College of Business Administration at Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia. He received B.Sc. from Thamar University, Yemen and M.Sc. and Ph.D. degrees from King Saud University, Saudi Arabia. He also served as faculty member of department of Computer Science at Taiz University, Yemen. His research interests include optimization and swarm intelligence.

Fahad Ahmed Khokhar received the B.S degree in Software Engineering, in 2018, and the M.S. degree in Computer Science, in 2016, from the COMSATS University, Islamabad, Pakistan. He is currently pursuing the Ph.D. degree with the University of Florence, Italy. From 2020 to 2022, he worked as a Research Assistant with the Department of Computer Science, COMSATS University, Islamabad, Pakistan. His research interests include anomaly detection, security and safety, Computer Vision, and implement appropriate architectures or software in the domain of critical systems.

Rubina Ghazal is serving as Assistant Professor at University Institute of Information Technology (UIIT), PMAS Arid Agriculture University Rawalpindi, Pakistan since 2006. She also served as Lecturer of Computer Science in University of Agriculture Faisalabad (UAF) from 2002 to 2005. She received her Ph.D. in computer science from COMSATS University Islamabad (CUI), Pakistan. She also holds M.S. and M.Sc. degrees in computer science from University of Agriculture Faisalabad, Pakistan. Her research interests include machine learning, cyber security, multi-agent systems, and semantic web technologies.