scientific data

Check for updates

OPEN

DATA DESCRIPTOR

E-POSE: A Large Scale Event Camera Dataset for Object Pose Estimation

Oussama Abdul Hay[1], Xiaoqian Huang[1], Abdulla Ayyad[1], Eslam Sherif[1], Randa Almadhoun[2], Yusra Abdulrahman[1,3], Lakmal Seneviratne[4,5], Abdulqader Abusafieh[1,6] & Yahya Zweiri[1,3] ✉

Robotic automation requires precise object pose estimation for effective grasping and manipulation. With their high dynamic range and temporal resolution, event-based cameras offer a promising alternative to conventional cameras. Despite their success in tracking, segmentation, classification, obstacle avoidance, and navigation, their use for 6D object pose estimation is relatively unexplored due to the lack of datasets. This paper introduces an extensive dataset based on Yale-CMU-Berkeley (YCB) objects, including event packets with associated poses, spike images, masks, 3D bounding box coordinates, segmented events, and a 3-channel event image for validation. Featuring 13 YCB objects, the dataset covers both cluttered and uncluttered scenes across 18 scenarios with varying speeds and illumination. It contains 306 sequences, totaling over an hour and around 1.5 billion events, making it the largest and most diverse event-based dataset for object pose estimation. This resource aims to support researchers in developing and testing object pose estimation algorithms and solutions.

Background & Summary

Object pose estimation is essential for a wide range of robot grasping and manipulation applications such as augmented reality[1] and warehouse automation[2]. Robotic arms with grippers and visual sensors can work tirelessly, handling diverse items and increasing throughput compared to human operators. One of the driving elements of such technologies was the Amazon Picking Challenge (APC)[3], which allowed industry experts and research teams worldwide to work together to advance the applicability of robots in the industrial setting. Conventional vision sensors, whether that is RGB cameras or RGB-D cameras[4–6] are commonly used as the perception sensor with data-driven methods being the state of the art in terms of accuracy and performance.

Data-driven methods have advanced beyond traditional approaches by learning to map images of objects directly to a 6D pose. This shift created a demand for datasets to train models under various scenarios and conditions. Early datasets for 6D pose estimation include LINEMOD and LINEMOD-OCCLUDED[7,8], contain 15 objects and over 1100 frames captured under different lighting conditions. T-LESS[9] focuses on texture-less objects, providing industry-relevant examples to extract 3D feature descriptors for 6D pose estimation. The YCB dataset[10] benchmarks everyday objects, offering standard metrics for manipulation tasks. YCB-Video[5] uses a subset of the YCB objects, featuring 21 objects and 92 videos, with approximately 1000 annotated images per object for 6D pose estimation. Table 1 summarizes common datasets and their specific attributes. These datasets remain central to advancing 6D pose estimation. However, conventional sensors often require structured environments, relying on controlled lighting[11] and slower operational speeds[12] for accurate pose estimation.

To address the limitations of speed and light conditions of conventional sensors, researchers are turning to Neuromorphic Vision Sensors (NVS), also known as event-based cameras. These bio-inspired sensors detect scene changes with high dynamic range, responding to positive or negative illumination variations[13]. This allows them to operate effectively under various lighting conditions without specific exposure settings. Unlike traditional cameras, which have limitations in tracking fast-moving objects due to shutter speed constraints[14],

[1]Advanced Research and Innovation Center (ARIC), Khalifa University of Science and Technology, Abu Dhabi, UAE. [2]Faculty of Technology, Department of Computer Science, University of Sunderland, Sunderland, UK. [3]Department of Aerospace Engineering, Khalifa University of Science and Technology, Abu Dhabi, UAE. [4]Khalifa University Center for Autonomous Robotic Systems (KUCARS), Khalifa University of Science and Technology, Abu Dhabi, UAE. [5]Department of Mechanical Engineering, Khalifa University of Science and Technology, Abu Dhabi, UAE. [6]Research and Development, Strata Manufacturing PJSC, Al Ain, 86519, United Arab Emirates. ✉e-mail: yahya.zweiri@ku.ac.ae

[HTML]C0C0C0 Dataset Name	Size (frames)	Cluttered	Occlusion	Varying light conditions	Motion blur	Texture-less Objects
LINEMOD (LM)[7]	18,273	✓	✗	✓	✗	✓
LINEMOD OCCLUDED (LM-O)[8]	1,214	✓	✓	✓	✗	✓
YCB-Video Dataset (YCB-V)[5]	133,827	✓	✓	✗	✗	✗
T-LESS[9]	47,664	✓	✓	✓	✗	✓
ITODD[43]	3,500	✓	✓	✗	✗	✓
TUD-L[29]	23,914	✗	✗	✓	✗	✗
TYO-L[29]	1,680	✗	✗	✓	✗	✗
HOPE[44]	2,038	✓	✓	✓	✗	✗
YCBInEOAT[45]	7,449	✗	✓	✗	✗	✓
[HTML]E6F7E6 **YCB-EV**[26]	13,851	✓	✓	✓	✓	✗
[HTML]E6F7E6 **RGB-D-E**[12]	2,500	✗	✗	✗	✓	✗
[HTML]E6F7E6 **E-POSE (ours)**	333,357	✓	✓	✓	✓	✗

Table 1. Comparison with existing datasets such as LINEMOD[7], LINEMOD OCCLUDED[8], T-LESS[9], ITODD, TUD-L[43], TYO-L[29], HOPE[44], YCBInEOAT[45] and two available event dataset YCB-EV[26] and RGB-D-E[12]. The attributes addressed in all datasets are summarized.

event-based cameras provide a continuous stream of micro-second level events[15], enhancing precision. Additionally, event-based cameras capture only events from moving objects, reducing bandwidth and improving the performance of downstream algorithms compared to conventional high-resolution imaging sensors[12]. This biologically-inspired sensor and neuromorphic hardware are anticipated to increase computation performance, especially with the end of Mooreâ€™s law approaching[16]. Neuromorphic processors operate with high parallelism, as all neurons and synapses can function simultaneously, performing simple computations. These systems integrate processing and memory, eliminating the separation found in traditional computing, which helps speed up operations and reduces energy usage[17]. Additionally, they use event-driven computation, operating efficiently and consuming less energy by processing only when necessary. Therefore, due to the uniqueness of the event-based camera modality, there is a need to develop a new class of hand-crafted features that could be used for event streams. Event-based cameras have already been employed in some applications such as countersink inspection[18], slip detection[19,20], and for grasping and manipulation[21–25].

However, to the best of the authors' knowledge, there are only two limited datasets of event-based cameras for 6DoF object pose estimation YCB-EV[26] and RGB-D-E[12]. YCB-EV[26] aimed to recreate the sequences in the original YCB-V dataset, while RGB-D-E[12] created annotations for one object for the total duration of 100 seconds. As shown in Table 1, YCB-EV and RGB-D-E contain 13,851 and 2,500 frames, respectively, whereas E-POSE provides 333,357 frames. This large number of frames is a result of the extensive number of sequences done along with our event frame generation approach, which depends on time intervals or event counts. Additionally, we provide an automated annotation script that allows users to generate data tailored to any applications requiring time surfaces, voxel grids, or accumulated frames. Since the event stream is continuous, the annotation script allows sampling the stream to create event frames and annotating them with the required frequency by the user. On the other hand, prior works annotated event frames by synchronizing them with RGB-D cameras. The dataset in[26] was annotated at a rate of 33 ms using an RGB-based network. However, the approach in[26] introduced limitations, such as throttling event data and potential annotation errors due to dropped RGB frames. The developed dataset in this work comprises 306 sequences with a total duration of more than 1 hour, making it the only large-scale event-based dataset encompassing many different scenarios and conditions for object pose estimation. In this paper, we provide a script that can discretize the raw event streams based on accumulation time or number of events and associate them with the poses, along with providing the segmentation masks, segmented events, and 3D bounding boxes at the appropriate interval for the user, since different applications would require different discretization needs. A benchmark is also provided based on three state-of-the-art networks PoseCNN[5], YOLO-6D[27], DGECN[28] to showcase the usage of the dataset to train a model.

Method

Experimental Setup. The hardware setup consists of two cameras mounted on a UR10 robot manipulator, which move them relative to the object. The two cameras used in this work were the DAVIS346c, which has a resolution of (346×260, and the ZED mini camera, which would provide the point cloud data to reconstruct the object and find its relative pose to the base frame $\mathcal{F}_{base}$. The UR10 is a highly accurate manipulator with a repeatability of 0.1 mm. The ZED mini camera used is stereo-based, the left camera out of the two cameras in the ZED mini camera was taken as the origin of the zed camera reference frame $\mathcal{F}_{zed}$. The setup of the two cameras and the UR10 robot is shown in Fig. 1. The objects used in this dataset are shown in Fig. 3. The selected objects were rich in features to generate as many events as possible; some selected objects were tools such as a wrench, a hammer, and scissors to diversify the range of objects in the dataset. Metallic objects are known for their reflective and featureless surface, which can hinder RGB methods due to the illumination affecting their features. Since event-based cameras focus on object edges, the illumination variation would not cause a significant change in the object perception. It would allow for more consistent feature extraction in different illumination conditions.

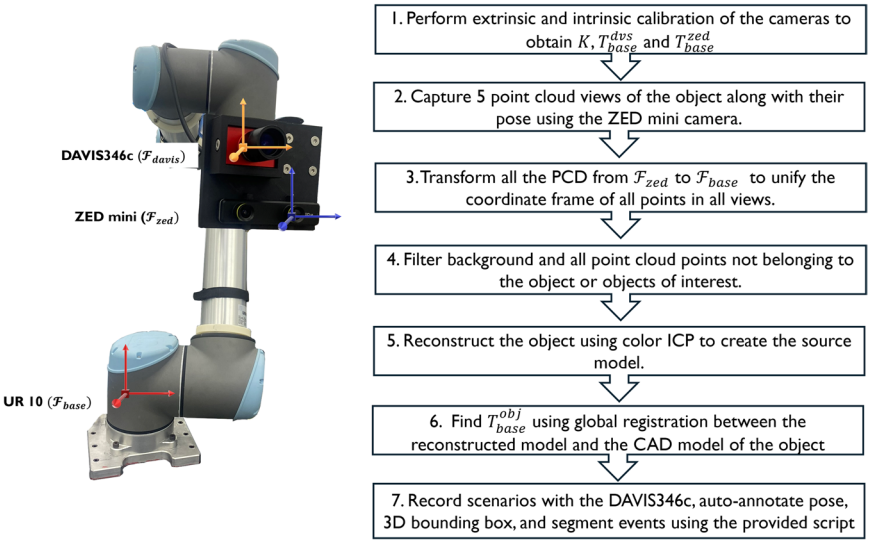


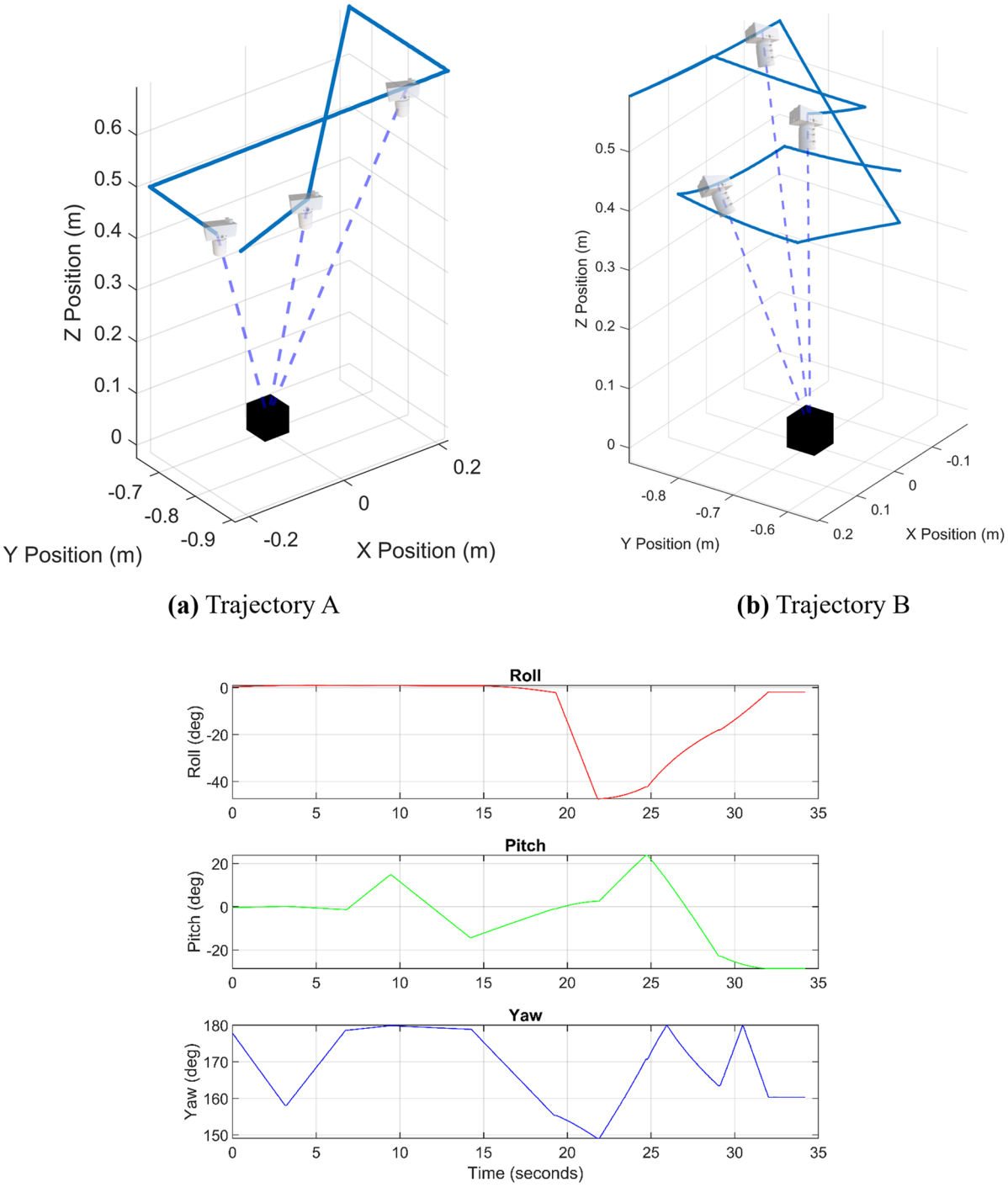
Fig. 1 Experimental hardware setup with the Event-based camera DAVIS346c $\mathcal{F}_{davis}$ and the ZED mini $\mathcal{F}_{zed}$ and the UR10 arm $\mathcal{F}_{base}$ along with their coordinate frame location, and overview of annotation method.

Experimental protocol. The dataset was collected for a subset of the YCB object models for events-based object pose estimation. After obtaining the pose, it is feasible to automatically segment the events and annotate the object's bounding box in the frames using standard 3D to 2D projection operations. 13 singular objects were considered, each appearing individually in the scene. Additionally, both cluttered and uncluttered scenes were considered in the data collection scenarios. In the uncluttered scenes, three objects were placed well-distanced from each other to facilitate the estimation of their poses without occlusion. In contrast, the cluttered scenes involved placing objects closer together to create occlusions during pose estimation. This process was repeated with five objects, arranging them both in well-spaced (uncluttered) and closely packed (cluttered) configurations. To introduce further variability to the dataset, the camera movement was varied. This was done by generating two main trajectories while recording a trajectory that only involves translation (**Trajectory A**) Fig. 2a across X, Y, and Z axes, and a trajectory with both translation and rotation (**Trajectory B**) Fig. 2b which was carefully devised to have the objects at all times within the field of view of the camera. The UR10 robot is controlled through high-level guidance using a linux computer. On this computer, a Python script computes the desired Tool Center Point (TCP) pose $T_{desired}$ for the robot based on the task requirements. The robot was controlled based on predetermined waypoints. These waypoints were chosen to span the SE(3) space of the object's relative location with respect to the DAVIS346c camera, ensuring that the object remained in the center of the camera's field of view. Communication between the linux computer and the UR10 controller is established using the real time data exchange (RTDEl) interface/API (https://sdurobotics.gitlab.io/ur_rtde/introduction/introduction.html) which allows real-time transmission of the desired pose to the robot. The UR10 controller then processes this input by calculating the inverse kinematics to determine the required joint angles and generating a smooth joint trajectory. Finally, the controller executes the movement, ensuring precise and accurate manipulation according to the commanded instructions. The different angle ranges are shown in Fig. 2c.

Furthermore, the object illumination was varied during data capture, similar to TUD-L and TYO-L[29]. This work aimed to sample different regions of the event-based camera's dynamic range. The work in[30] defines four illumination categories: Bright Urban Center (200–2000 lux), Normal Street Lighting (20–200 lux), Low Street Lighting (2–20 lux), and Moonlight (0–2 lux). In this work, we focused on the top three categories, as moonlight conditions often result in poorly defined edges and corners, reducing the accuracy of object detection and pose estimation. To achieve this, data was captured in a controlled environment with three lighting conditions: good light (830 lux), moderate light (170 lux), and low light (18 lux).

For each lighting condition, the speed of the UR10 robotic arm was varied to induce motion blur as an additional attribute of the dataset. Dynamic object grasping, although less explored than static object handling, is crucial for robotic automation. Prior works, such as[31], assessed grasping success at speeds between 0.05 m/s and 0.5 m/s. Similarly, the ESD dataset[21] demonstrated that the performance of RGB networks declines as the robot speed increases from 0.15 m/s to 1 m/s due to motion blur. Therefore, we evaluated the performance of the event camera under varying speeds similar to the ones commonly used in the community to demonstrate its applicability in dynamic and high-speed environments.

The maximum speeds reached in (**Trajectory A**) and (**Trajectory B**) during data capturing were 0.1 m/s, 0.5 m/s, and 1 m/s. In all these scenarios, event, RGB, and point cloud data were recorded. It is noted that in (**Trajectory A and B**), a more random motion rather than a structured one (such as a circle or square) was introduced to limit over-fitting when training the data since a more structured trajectory might hinder the model's ability to infer the pose based on the captured shape of the object. A similar path with a reduced span was generated for the multi-object scenarios to ensure all objects were within the field of view during movement. Both cameras' intrinsic and extrinsic parameters were obtained through standard calibration procedures.



(a) Trajectory A

(b) Trajectory B

(c) Roll, Pitch and Yaw angles variations during Trajectory B

Fig. 2 Figure (**a**) and Figure (**b**) illustrate the two different trajectories considered during data collection.

Calibration. Intrinsic and extrinsic calibrations are one of the fundamental blocks of using vision in robotics. The accuracy of the calibration would dictate the accuracy of the data annotation later done. For the intrinsic calibration of the DAVIS346c and the ZED mini, the function from Opencv *calibrateCameraCharuco* was used to obtain the intrinsic matrix and the distortion coefficients of the two perception sensors. A re-projection error of 0.3 and 0.5 pixels were obtained for the ZED mini and the DAVIS346c, respectively. The extrinsic calibration is important to obtain T_{base}^{zed} and T_{base}^{dvs}, which are necessary for 3D reconstruction of the object and annotation of the pose along the trajectory of the UR10. The transformations were estimated using a Hand-Eye calibration procedure that was done in[32]. An aruco checkerboard with known dimensions that enable us to obtain the transformation from the camera frames $\mathcal{F}_{dvs}$ and $\mathcal{F}_{zed}$ to the aruco checkerboard frame $\mathcal{F}_{aruco}$ was used. A system of

Fig. 3 Subset of the YCB dataset used for creating E-POSE. The Objects that were chosen were 1- Wood block 2- Sugar box 3- Bleach cleanser 4- Scissors 5- Drill 6- Hammer 7- Wrench 8- Potted meat can 9- Cracker box 10- Rubik's cube 11- Nine hole peg test 12- Timer 13- Mustard bottle.

homogeneous matrix equations $AX = ZB$ is used in an optimization scheme by obtaining multiple viewpoints of A and B and then solving for X and Z in a nonlinear constrained minimization process as outlined in[32]. The matrices in $AX = ZB$ are:

- A is the transformation from $\mathcal{F}_{zed}$ to $\mathcal{F}_{aruco}$ or from $\mathcal{F}_{dvs}$ to $\mathcal{F}_{aruco}$ in case of the event camera.
- X is the transformation from the end-effector of the UR10 robot to the camera frames $\mathcal{F}_{zed}$ and $\mathcal{F}_{dvs}$.
- Z is the transformation from the $\mathcal{F}_{base}$ to $\mathcal{F}_{aruco}$.
- B is the transformation from the end-effector to $\mathcal{F}_{base}$.

The optimization, when done separately for each sensor, converges to an RMSE less than 10^{-9}, which indicates a high accuracy of the T_{base}^{zed} and T_{base}^{dvs} obtained.

Furthermore, the temporal calibration is as imperative as the spatial calibration. The temporal calibration ensures that the data points used by the robot and the DAVIS346c camera are synchronized. As done in[33], a property of the event camera was used, which is as the event camera's velocity increases, the changes in the intensity values of the pixels increase, and the number of events generated increases. Since the DAVIS346c was mounted on the UR10 robot, the velocity of the UR10 and the event camera are similar. By directly matching the minima between the UR10 TCP velocity along with the number of events available in each frame, the minimum number of events would correspond to the local minima in the velocity of the robot. After finding the time discrepancy and synchronizing both streams, the accuracy of these methods would rely on the time cycle for each sensor independently. However, since the robot and the sensor operate on separate machines, the time offset between the two data streams is expected to remain consistent over the short duration of the sequence. Moreover, the inaccuracy associated with this method tends to increase over longer sequences due to clock drift. However, for the short sequences collected here, it would be minimal.

3D Reconstruction. To obtain the ground truth 6D pose of the object, the ZED mini camera was used to reconstruct the objects using a point cloud from multiple views. The multiple views captured the different sides of the object on the table in front of the manipulator. The point clouds from 5 different perspectives were captured along with the pose of the ZED mini camera during capturing. The 5 point clouds were then transformed from the frame of the camera $\mathcal{F}_{zed}$ to UR10 base frame $\mathcal{F}_{base}$ using the extrinsics of the cameras $\mathcal{F}_{zed}$ relative to the $\mathcal{F}_{base}$. This is done to have a uniform fixed frame for all point clouds when fitting them with each other. The point cloud files consist of each point's x,y, and z coordinates, along with its RGB components. To transform the reference frame from $\mathcal{F}_{zed}$ to $\mathcal{F}_{base}$, Eq.1 was used, where pc_i^{zed} denotes all the points in the point cloud with the reference of $\mathcal{F}_{zed}$, and T_{base}^{zed} is the transformation between the $\mathcal{F}_{base}$ and the ZED mini camera $\mathcal{F}_{zed}$.

$$pc_i^{base} = pc_i^{zed} \times T_{base}^{zed} \tag{1}$$

After transforming all point clouds pc_i^{zed} to $\mathcal{F}_{base}$, both color and geometry[34] of the point clouds were used for registration, the inclusion of color data stabilizes the alignment across the tangent plane, enhancing the accuracy and robustness of this algorithm compared to previous point cloud registration methods. Additionally, its execution speed remains on par with ICP (Iterative Closest Point) registration. The algorithm in Eq. 2 optimizes two objective functions, a photometric part and a geometric part, Where δ is a weighing term between [0,1].

$$E(T) = (1 - \delta)E_C(T) + \delta E_G(T) \tag{2}$$

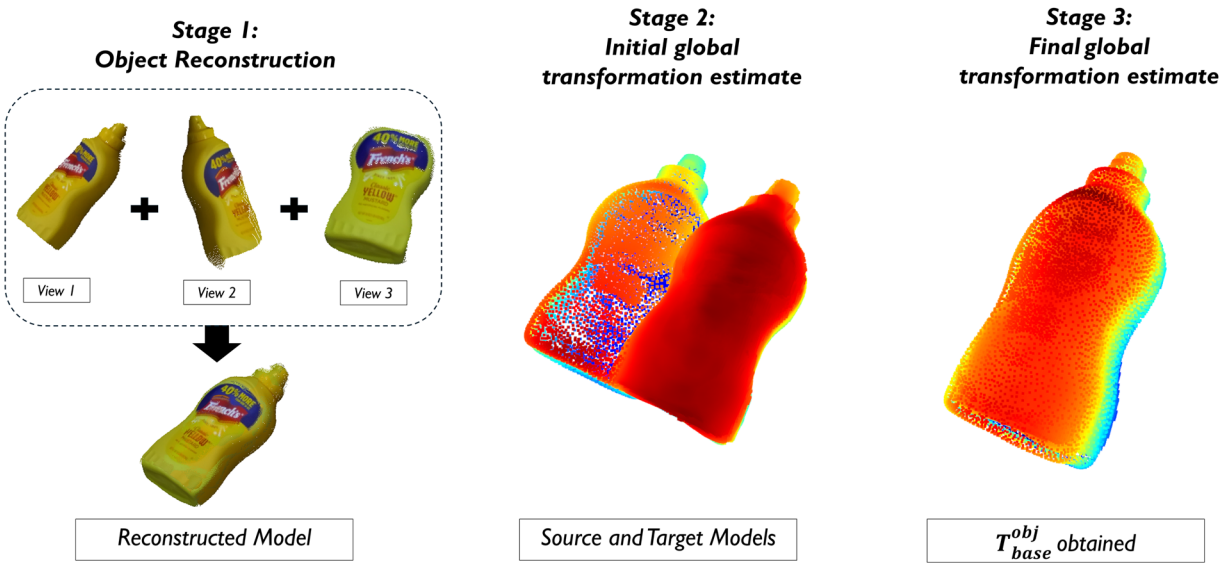


Fig. 4 Stage 1 in the figure illustrates the 3D reconstruction procedure to create the object from the collected views. The reconstructed model is used along with the CAD model to give an initial estimate of T_{obj}^{base} to allow the global ICP algorithm to converge faster. Stage 3 shows the final result with the CAD model superimposed over the reconstructed model, and T_{obj}^{base} is obtained.

$E_G(T)$ is the point-to-plane ICP algorithm by[35], the point-to-plane ICP algorithm aims to minimize the perpendicular distances between points in one cloud and the planes tangent to corresponding points in the other cloud. This is achieved by using point normals for the calculation. This technique typically results in faster convergence speeds compared to the point-to-point ICP.

$$E_G(T) = \sum_{(p,q)\in\mathcal{K}} ((\mathbf{p} - T\mathbf{q}) \cdot \mathbf{n}_p)^2 \tag{3}$$

In Eq. 3 above (p, q) are the source and target point clouds, $\mathcal{K}$ is the set of correspondence. and n_p is the normal vector of p.

$$E_C(T) = \sum_{(p,q)\in\mathcal{K}} (C_p(f(T\mathbf{q})) - C(q))^2 \tag{4}$$

In Eq. 4, $E_C(T)$ is the difference between the colors of the source point clouds denoted by $C(q)$ and its projection on the tangent plane p. The colored registration is done over multiple scales of voxels to find the one that optimizes the correspondence distances. The model is fitted with the laser-scanned model provided by the dataset using point-to-point ICP in Eq. 5 to find the global pose of the object relative to the UR10 base. The registration process was conducted through multiple iterations to achieve high precision, a maximum RMSE for the registration of all objects was $3\,e^{-3}$ m. The output of the registration was a 4×4 transformation matrix T_{base}^{obj}. The different steps completed for 3D reconstruction is summarized in Fig. 4.

$$E(T) = \sum_{(p,q)\in\mathcal{K}} ||\mathbf{p} - T\mathbf{q}||^2 \tag{5}$$

Events annotations. After obtaining the ground truth pose T_{base}^{obj} of the object relative to the $\mathcal{F}_{base}$, the task was then to associate the events with the poses as the camera traverses the trajectories in Fig. 2a and b. The two streams, events and robot poses are adjusted for the time discrepancy in the method mentioned in section *Calibration*. Following that, The script provided with the data allows the user to create event frames depending on the representation needed by the user **(Step 1)**. In this work, to evaluate the dataset, a time interval Δt of 0.01 s was selected, representing a frequency of 100 Hz. The time of the last event added to the frame was used as the time stamp of the frame. Linear interpolation and spherical linear interpolation (slerp) was used to obtain the pose of the camera along the predetermined trajectory of the robot **(Step 2)**. T_{base}^{obj} previously obtained is then used to find the pose of the object relative to the camera T_{dvs}^{obj} **(Step 3)**, the steps are illustrated in Fig. 5. The interpolation was performed locally and per dimension for the linear motion to ensure that the variations in the continuous trajectory followed by the camera were captured at all the discretized timestamps. Equation 6 shows the interpolation for the X-axis, which was similarly applied to the Y and Z axes.

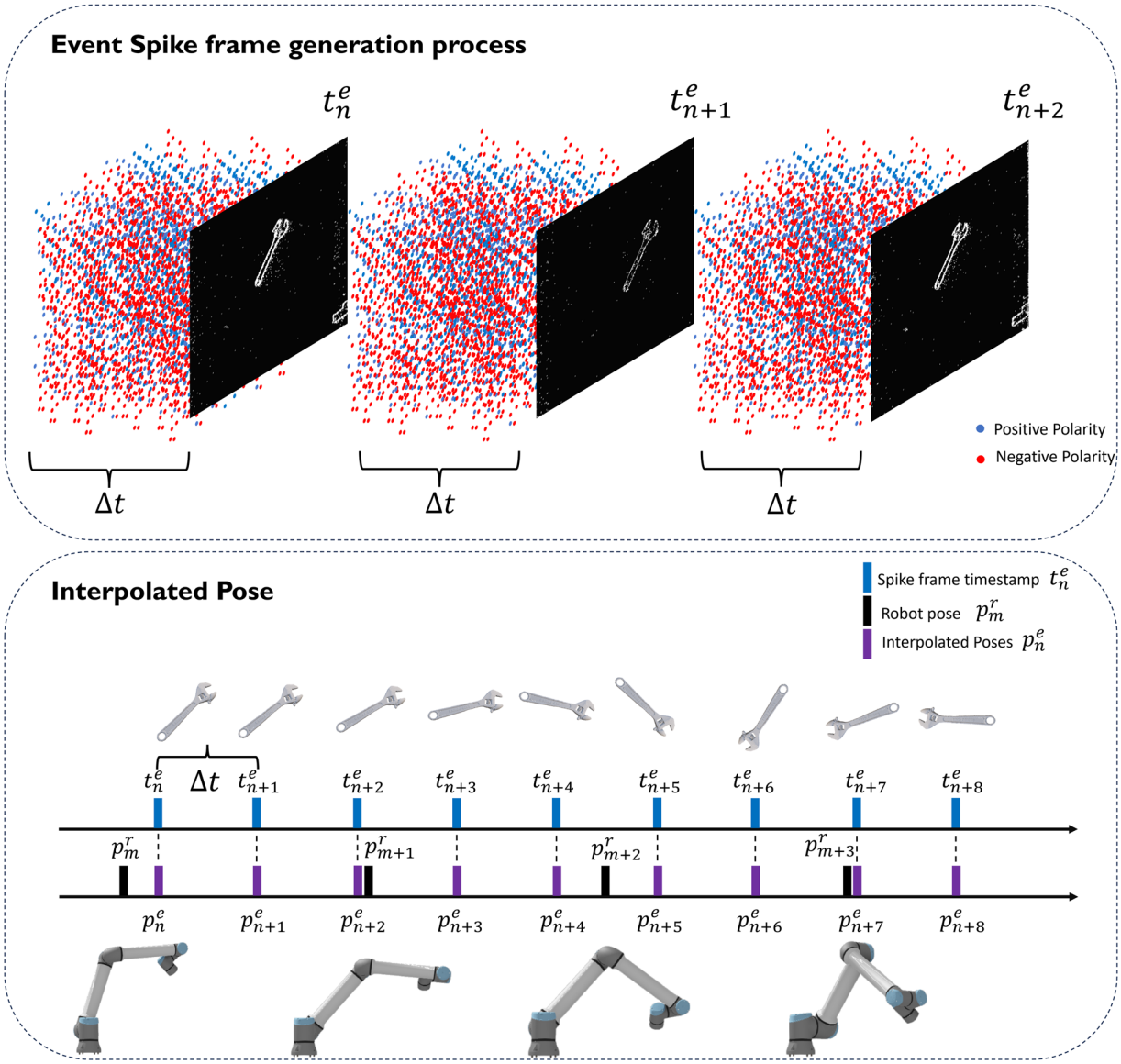


Fig. 5 The accumulation of events into frames is illustrated at specific time points: t_n^e, t_{n+1}^e, and t_{n+2}^e. These frame timestamps are then used as interpolation points to estimate the poses p_e^n, based on the poses p_r^m obtained from the robot. The wrench depicts the change in the pose of the object at different times of the event frames, and the robot depicts different poses along the trajectory to be interpolated from.

$$X(t^N) = X(t^{N-1}) + \frac{t^N - t^{N-1}}{t^{N-2} - t^{N-1}}(X(t^{N+1}) - X(t^{N-1})) \tag{6}$$

Moreover, regarding rotations, *slerp* was used on the normalized quaternions to estimate the rotation at the discretized times. The equation for slerp as shown in Eq. 7 where $q(t^N)$, $q(t^{N-1})$, and $q(t^{N+1})$ are the normalized quaternions for the current, previous and future time stamps respectively. θ is half the angular distance between $q(t^{N-1})$, $q(t^{N+1})$ and T is a scalar that determines how close the current quaternion is to either the previous or the future iterations. The proximity to current and previous time stamps is determined using the time stamps associated with each pose.

$$q(t^N) = \frac{sin\,((1 - T)\theta}{sin(\theta)} q(t^{N-1}) + \frac{sin(T\theta)}{sin(\theta)} q(t^{N+1}) \tag{7}$$

After completing the interpolation process, the discretized events are projected onto a frame of 346×260 to create a spike frame. The spike frame is adjusted for distortions using the intrinsic matrix K and the distortion coefficients D obtained during the intrinsic calibration procedure.

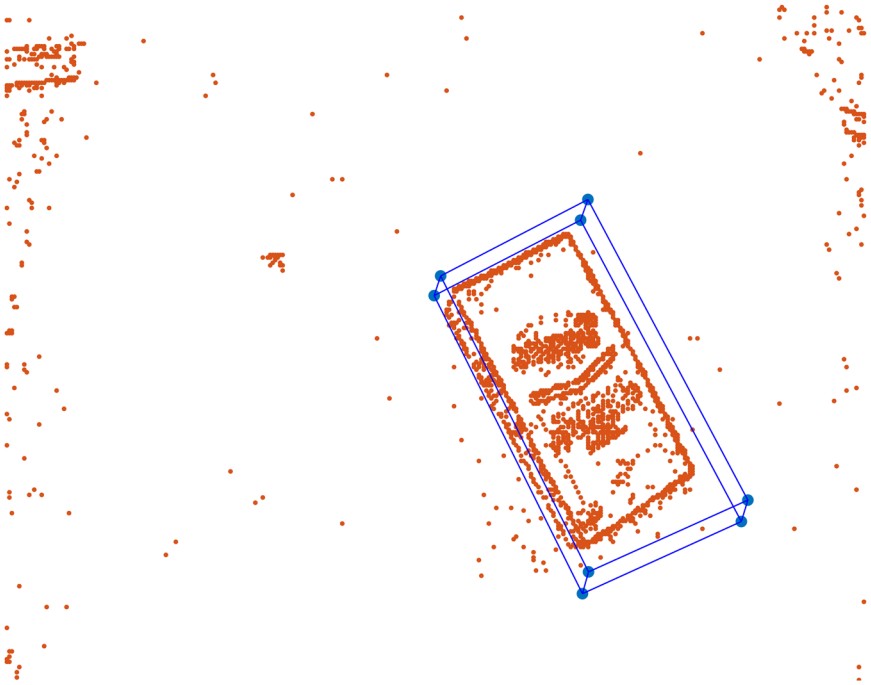


Fig. 6 Sugar box object along with its projected 3D bounding box.

After creating the spike frames by accumulating events based on the number of events or a specific accumulation time, the poses that correspond to those frames at each instant t_n^e is interpolated from the original trajectory stream to get p_n^e using the equation below Eq. 8. p_n^e is the transformation of the object relative to the camera $T_{dvs}^{obj}(t_n^e)$.

$$T_{dvs}^{obj}(t_n^e) = (T_{base}^{dvs}(t_n^e))^{-1} \times T_{base}^{obj} \tag{8}$$

For object pose estimation, it is also important to provide a 3D bounding box that encapsulates the object. In this work, since the CAD object model was available for all the objects used, the object model was utilized to obtain the 8 extremities of the object model in the object frame $\mathcal{F}_{obj}$. The 8 points w_1 to w_8 are 3D points that combine the permutations of the maximum and minimum of all 3 dimensions of the CAD model. A small user-defined margin ε was added to ensure that the object fits completely within the 3D bounding box. To depict the 3D bounding box in the frame, the 8 points of the 3D bounding box were transformed to the image plane. This transformation utilized the pose obtained p_n^e to transform the points from the object frame $\mathcal{F}_{obj}$ to the camera frame $\mathcal{F}_{dvs}$, and the intrinsic matrix K to project the points onto the image.

$$3Dbbox_i^{world} = T_{dvs}^{obj}(t_n^e) \cdot w_i \tag{9}$$

In Eq. 9 w_i is a 4×1 homogeneous coordinate vector of the form $[x_i, y_i, z_i, 1]$ that represents the location of the eight bounding box points in the $\mathcal{F}_{obj}$ frame. The points are then projected onto the image using perspective projection.

$$3Dbbox_i^{image} = \left[\frac{f_x \cdot 3Dbbox_i^{world}(1)}{3Dbbox_i^{world}(3)} + c_x, \frac{f_y \cdot 3Dbbox_i^{world}(2)}{3Dbbox_i^{world}(3)} + c_y \right] \tag{10}$$

In Eq. 10 f_x, f_y, c_x *and* c_y are the focal lengths and principal points in the x,y image coordinates respectively. The results of the 3D bounding box projections can be seen in the Fig. 6.

Additionally, the segmentation and mask images for the spike frame images I were prepared. To obtain the mask M and segmentation images S, which would contain the events that belong to the segmented object only, the CAD model, a point cloud scan of the object, is projected onto the image plane. The CAD model consists of millions of points, sampled from the object, that represent the object's surface. A million points were sampled from the object's CAD model before being projected to create a continuous mask.

$$U = K \cdot T_{dvs}^{obj} \cdot PC' \tag{11}$$

The generated image U is a 3-dimensional vector composed of the projection of the point clouds to x,y pixel locations and the depth of each point within the cloud.

$$M = \begin{cases} 1 & \text{if } U(1) > 0 \text{ or } U(2) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

To retain only the events corresponding to the object and effectively eliminate background noise associated with the event camera, the binary mask M, which indicates the pixels occupied by the object at a specific time instant in its trajectory, is applied to filter out and remove events outside the object region. The segmented image S is obtained by multiplying element-wise the mask M and the spike frame I. The results can be shown in Fig. 7, where the drill CAD is first projected onto the frame and then used to segment the events in the spike image.

Furthermore, while this method performs effectively for single objects, it encounters challenges in cluttered scenes where objects overlap, and multiple objects may occupy the same pixel. In such cases, the third dimension of the projected point cloud, represented by $U(3)$, indicates the proximity of each point to the camera. By selecting the projection with the smallest $U(3)$ value in overlapping pixels, the mask of the object closest to the camera can be segmented. Moreover, the projection of the mask onto the spike image to segment the events, as seen in Fig. 7, provides a visual feedback of the accuracy of the pose at all instances by utilizing the obtained pose to conduct the segmentation process.

Algorithm 1 6D pose Automatic Annotation for Events Data.

Input: Events stream: $E = \{(x, y), p, t_{se}\}$, Camera Pose: p;
Fix time shift of poses and events by matching velocity and number of events peaks.
Sync the start time of p_1^r based on Event time t_1^e;
Divide the event steam to event windows based Δt to end up with N spike frames;
for $j = 1 : N$ **the number of event frames do**
    Find the local poses before p_{t-1}^r and after p_{t+1}^r the event frame I_j
    Find the scale for which the t of event frame is close to time of poses $t-1$ and $t+1$
    Use spherical linear interpolation to interpolate the quaternion at that particular time t, slerp(p_{t-1}^r,p_{t+1}^r,$scale$)
    Use linear interpolation to find the translation (x,y,z) at t_j
    Combine the rotation and translation to one matrix T_{base}^{dvs}
    Find T_{dvs}^{obj} for all poses p_n^e using Eq. 8.
    Create the three-channel image using positive and negative polarities and the latest time stamp channels.
    Undistort the spike and three-channel image
    Find the location of the 3D bounding box using Eq. 10.
    Create the mask image M and segment the events belonging to the object in the spike frame.
Write all data to file.

Data visualizations. The dataset was extracted and organized into images, text, and MAT files containing the necessary information. Each data file is named according to the scenario it follows. The first letter, `r` or `t`, indicates whether the trajectory contains only translations or includes rotations. The numbers `01`, `05`, and `10` denote the speed of operation of the robot (0.1 m/s, 0.5 m/s, and 1.0 m/s, respectively). The last two letters indicate the lighting conditions: `ll` for low light, `ml` for moderate light, and `gl` for good light.

For all single-object scenarios, the dataset includes:

- Event spike image (I)
- Mask image (M)
- Segmented event spike frame (S)
- Three-channel image
- 3D bounding box in pixel coordinates text file
- Path to all images
- 3D bounding box in world coordinates in the object frame $\mathcal{F}_{obj}$ text file

For multiple object folders, the text files containing the 3D bounding box world coordinates have n rows (where n indicates the number of objects) and eight columns providing the eight bounding box coordinates of the object in the object frame.

For multiple-object scenarios, the dataset includes everything provided for single-object scenarios, along with a segmentation mask and segmented events image for each object in the scene. This will enable users to utilize the dataset for future event-based segmentation applications.

The MAT file provided with each image contains the information necessary to train an object pose estimation network. Similar standards were followed with the YCB-Video dataset[5]. The information provided is:

- **centerInfo**: the 2D location of the center of the 3D model of the object in the image
- **Pose**: 4×4 transformation matrix of the pose of the object relative to the camera T_{dvs}^{obj}
- **IntrinsicMatrix**: camera intrinsics of the DAVIS346c
- **cameraPose**: Pose of the camera relative to the UR base T_{base}^{dvs}.
- **time**: Time of the current pose and frame.
- **keypointprojections**: 2D and 3D locations of the 3D bounding box in the camera frame.

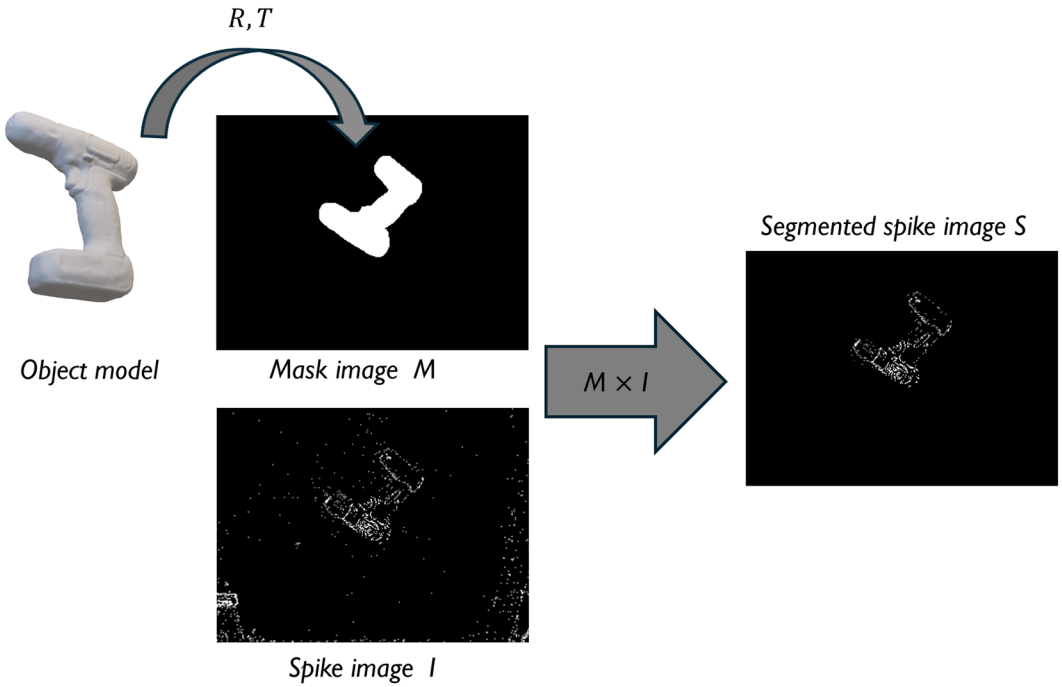


Fig. 7 3D to 2D projection of the CAD model and segmenting of the events belonging to the object using the obtained T_{dvs}^{obj}.

- **events**: All the events used to construct the images at that specific index with all the needed info $<x_k, y_k t_{se}, p_k>$ where x,y are the pixel location, t is the event time stamp and p is the polarity being +ve or -ve.
- **cls index**: Index of the class of the object.

Data Records

All data is currently available in[36], and the data structure is as seen in Fig. 8.

Data format. The data[36] provided originally is in '.h5' format. The data[36] was extracted and discretized to intervals based on Δt. The generation script ensures data extraction into '.png' images such as the spike image, mask, segmented events, and the three-channel image. In addition, all the data related to the pose is available in the '.mat' file provided alongside each sample image.

Dataset challenging factors and attributes. The dataset was constructed to introduce as much variability as possible in the indoor lab environment. Different scenarios were added that made the dataset rich with attributes. The total number of sequences and scenarios for each attribute is shown in Fig. 10a, Fig. 10b, and Fig. 10c. A total number of 306 sequences are available in this dataset for the 13 single objects and four multiple object scenes. The total number of event frames generated and events is 333 K and 1.53 billion, respectively. The different attributes available in the dataset are as follows:

- **Single Object (SO)**: In these scenarios, only one object was considered in the scene. The trajectory was designed to change the pose while the camera perceived one object from all different angles. The 3D bounding box, mask, and segmented events were automatically generated. There are 13 objects in this dataset, and each was used in 18 different conditions.
- **Multiple Objects (MO)**: Multiple objects were considered in these scenarios. The poses of all objects were captured, and from that, the 3D bounding box, mask, and segmented events were automatically generated. 4 different scenes contain multiple objects, each with 18 different sequences featuring various lighting conditions and speeds of traversal.
- **Cluttered Scene (CS)**: This scene included multiple objects placed in a cluttered manner. The overlapping objects added difficulty in determining their poses due to occlusion. Cluttered scenes with 3 and 5 objects were considered, each with 18 different sequences that varied the lighting and speed of traversal.
- **Uncluttered Scene (UCS)**: This scene included multiple objects spread out to avoid any object being obstructed by another. Scenarios with 3 and 5 objects were used in 18 sequences, where lighting and the robot's speed varied.
- **Low Light (LL)**: Data collection was conducted under low light while varying the speed. This was applied to all single-object and multiple-object scenes.

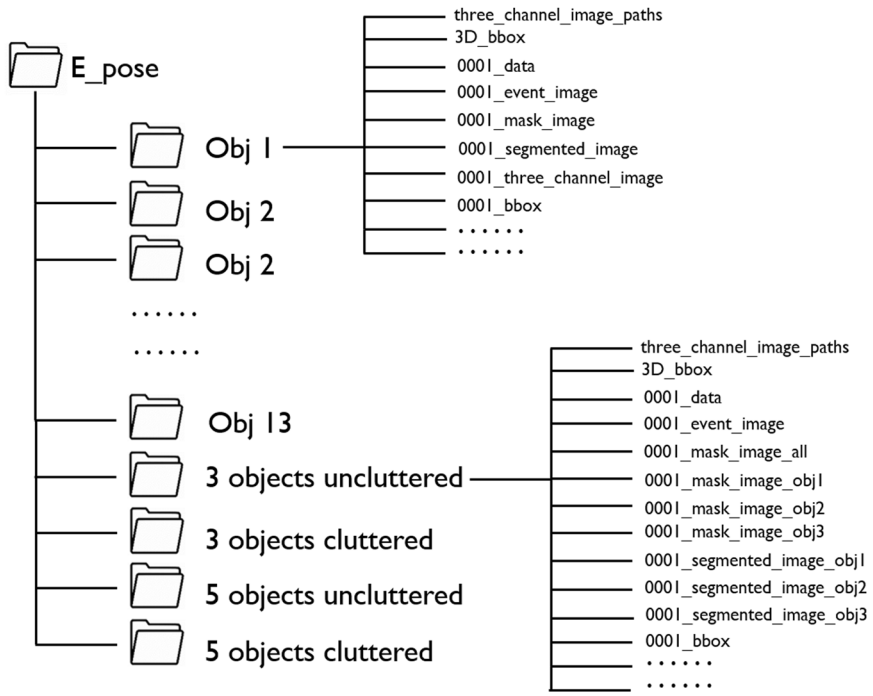


Fig. 8 Data structure, each Obj folder is named according to which object it belongs in the dataset[36]. Each Obj file will contain the images and pose information used and needed for object pose estimation. The multiple object folders also contain a mask for each object in the scene, along with the poses of each object.

- **Moderate Light (ML)**: Data collection was conducted under moderate lighting conditions while varying the speed. This was applied to all single-object and multiple-object scenes.
- **Good Light (GL)**: Data collection was conducted under good lighting conditions while varying the speed. This was applied to all single-object and multiple-object scenes.
- **Low Speed (LS)**: Data collection was conducted at low speeds (0.1 m/s) while varying the light conditions. This was applied to all single-object and multiple-object scenes.
- **Moderate Speed (MS)**: Data collection was conducted at moderate speeds (0.5 m/s) while varying the light conditions. This was applied to all single-object and multiple-object scenes.
- **High Speed (HS)**: Data collection was conducted at high speeds (1.0 m/s) while varying the light conditions. This was applied to all single-object and multiple-object scenes.
- **Translation Only (TO)**: This scenario utilized (**Trajectory A**) 2a during operation, with no rotational motion during data collection for all single-object and multiple-object scenarios.
- **Rotation and Translation (RO)**: This scenario utilised (**Trajectory B**) 2b during data collection for all single-object and multiple-object scenarios.

All the different scenarios for single and multiple objects can be depicted in Fig. 9.

Technical Validation

Evaluation metrics. The Average Distance of Model Points (ADD) metric, a widely adopted evaluation measure for 6D pose estimation, is used to estimate the accuracy of object pose estimation on our dataset via several state-of-the-art approaches. This metric assesses the accuracy of the predicted pose of an object by comparing the estimated transformation to the ground truth. Specifically, ADD computes the mean Euclidean distance between the corresponding 3D model points transformed by the ground truth pose and the predicted pose. For an object with n points, let $\mathcal{M} = \{\mathbf{pc}_i | i=1, \ldots, n\}$ denote the set of 3D model points, $\mathbf{R}$ and $\mathbf{t}$ be the ground truth rotation and translation, and $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ be the estimated rotation and translation. The ADD metric is formulated as follows:

$$\text{ADD} = \frac{1}{n}\sum_{i=1}^{n}||(\mathbf{R}\mathbf{p}_i + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{p}_i + \hat{\mathbf{t}})||$$

where $||\cdot||$ denotes the Euclidean distance. However, multiple poses can be indistinguishable in the visual data for symmetric objects, leading to multiple correct predictions. So, the ADD-S metric, a variant of the ADD metric designed to handle symmetric objects, is used in this paper. Considering the closest point distance, the ADD-S metric modifies the original ADD metric to account for these symmetries. It is defined as follows:

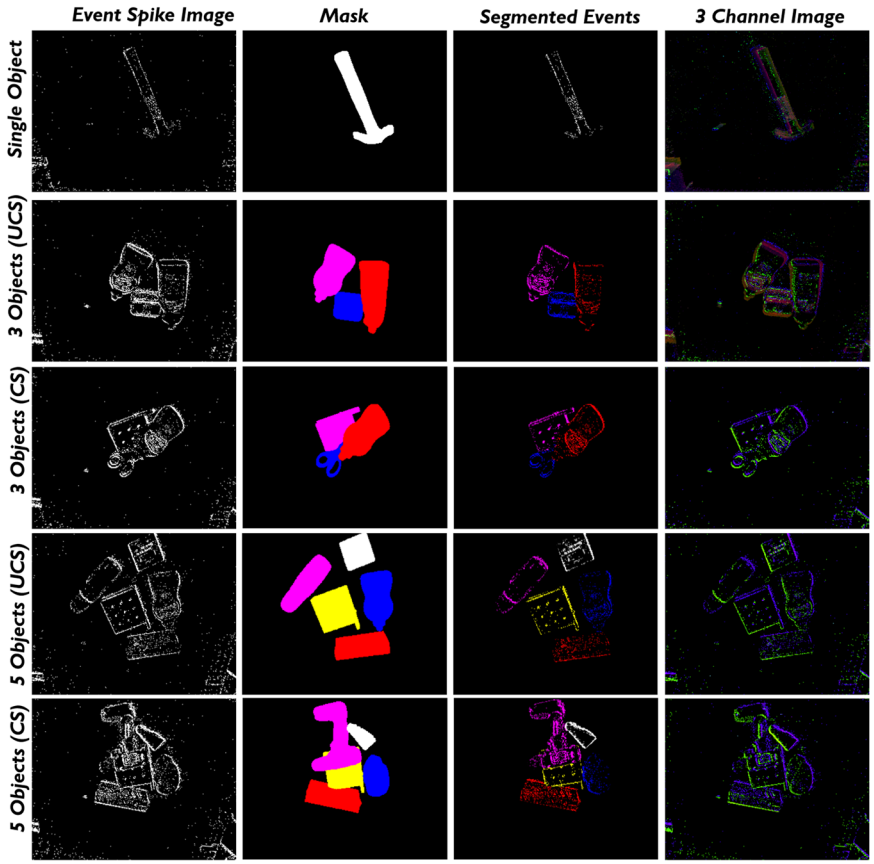


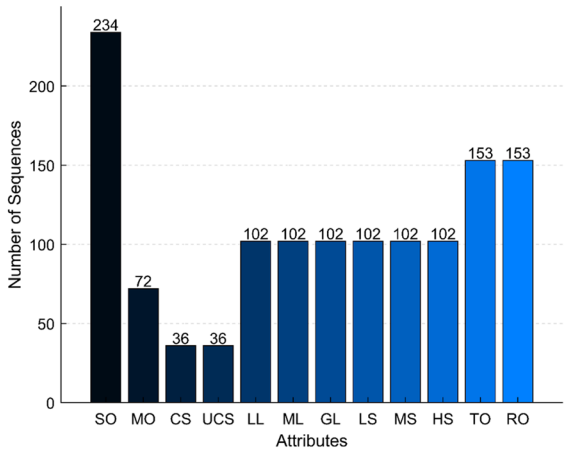
Fig. 9 Sample images of single object and multiple objects in the scene. The first column shows the event spike image, a 1-channel binary image. The second column shows the mask image obtained through the pose and CAD model of the objects. The third column shows the segmented events of each object, and the fourth column shows the three-channel images created to validate the dataset on pre-trained models.

$$\text{ADD} - \text{S} = \frac{1}{n}\sum_{i=1}^{n} min_{\mathbf{p}_j \in \mathcal{M}} \|(\mathbf{R}\mathbf{p}_i + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{p}_j + \hat{\mathbf{t}})\|$$

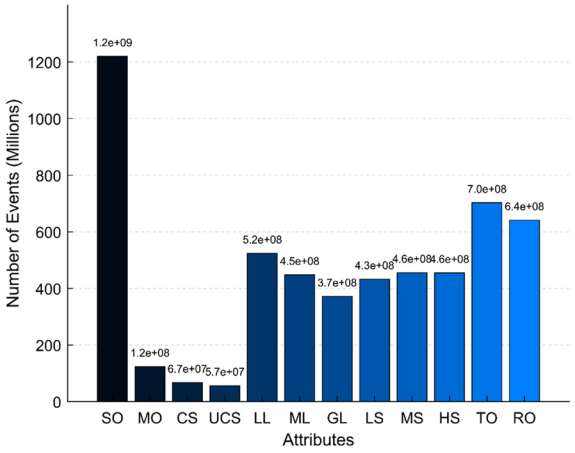
Pose estimation on event frames. As mentioned, to the best of our knowledge, there is no deep learning work addressing object pose estimation for solely event data. We convert event data into grid-like representations to leverage state-of-the-art approaches developed for conventional vision.

To test the event-based camera output on the image-based pre-trained networks, a representation must be developed with three channels (similar to RGB images) that would capture the scene and provide valuable information to the network to estimate the object pose. The event camera is inherently superior to frame-based cameras because it captures the time of each pixel or event change in the stream. This directly allows the encoding of fine-grained motion information that would help in many motion-related applications. This was shown in[37] that optical flow can be determined linearly from a local window around each event by estimating a plane in the spatio-temporal domain compared to the iterative process commonly followed in frame-based cameras.

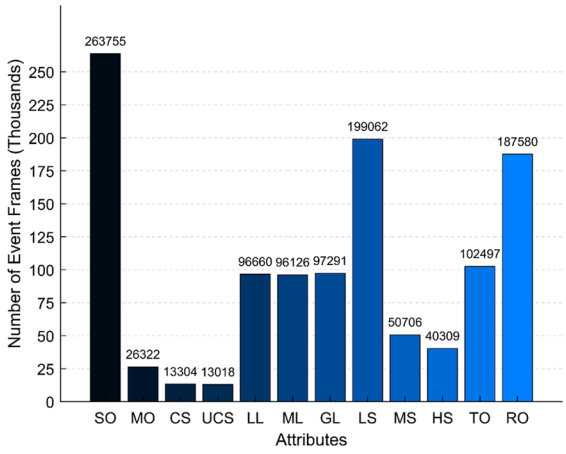
In this work, the following representation was similar to the one adopted in EV-FlowNet[38]. In that work, four channels were used to estimate the optical flow from an event stream. However, since only three channels were needed in this work, the first two channels were taken from EV-FlowNet[38], which are the counting of the positive polarity events and negative polarity events, respectively, in the discretized time window. The motivation is that event counting is a universal method for perceiving and visualizing the event stream. The technique was also used for 6Dof pose estimation of the camera in a learning-based approach in[39]. Moreover, counting the positive and negative polarities does not suffice, as argued in[38]. The absence of temporal information will lead to loss of information; therefore, the third channel selected represents the most recent timestamp of an event at a given pixel. This channel shows a gradient change in the intensity as the object's speed fluctuates, which is an advantageous feature when predicting the 6DoF. All three channels are normalized to ensure their maximum value is 1. This normalization helps maintain a consistent scale across the counting channels (positive and negative polarities) and the recent timestamp channel. In addition, this representation keeps the same magnitude for scenarios such as fast motion within a small time window and similarly slow motion within a large time window as mentioned in[38].



(a) Number of sequences for each attribute in the dataset



(b) Number of events for each attribute in the dataset



(c) Number of frames for each attribute in the dataset

Fig. 10 Figure (**a**), (**b**), and (**c**) illustrate quantitative data of the number of sequences, events, and frames generated for each attribute in the dataset.

Terms	Input	Pose DoF	ADD-S (<0.1 m) Ours (Event Frames)	ADD-S (<0.1 m) YCB-V[5] (RGB)	ADD-S (<0.1 m) LINEMOD[7] (RGB)
YOLO-6D[27]	RGB & CAD	6DoF	6.86%	—	56.00%
PoseCNN[5]	RGB & CAD	6DoF	15.78%	75.90%	—
DGECN[28]	RGB & CAD	6DoF	29.72%	90.90%	—

Table 2. Evaluation results of the state-of-the-art object pose estimation networks YOLO-6D[27], PoseCNN[5] and DGECN[28] on event frames of our dataset and RGB frames of public datasets YCB-V[5] and LINEMOD[7].

Existing conventional vision-based pose estimation approaches can be categorized as regression-based, template-based, and feature-based methods[40]. Among these methods, one of the straightforward approaches is to treat 6DoF object pose estimation as a regression task, directly predicting poses from the input RGB images without relying on intermediate keypoint representations. PoseCNN[5] uses direct regression to estimate the pose from RGB images, while DGECN[28] and YOLO-6D[27] rely on inferring correspondences and use PnP to obtain the pose. We applied transfer learning to PoseCNN and YOLO-6D on our dataset by unfreezing the last layer of the encoder, the whole decoder, and fully connected layers. However, DGECN is only partially open-sourced, which impedes the application of transfer learning on our datasets, thus direct inference was applied. Table 2 lists the quantitative testing results of object pose estimation. All the networks exhibit significant performance declines on our dataset when using event frame representations, compared to their testing results on public RGB frame-based dataset YCB-V[5] or LINEMOD[7].

When transferring or fine-tuning a network trained on RGB frames to event frames, the performance drop can be attributed to several key reasons. First, RGB frames capture absolute intensity values, whereas event frames capture changes in intensity. This fundamental difference means that the patterns and features learned by a network trained on RGB data might not directly apply to event data. Features such as color gradients and texture details in RGB frames are not available in event frames that focus on motion and changes. Besides, RGB frames provide rich spatial information, while event frames emphasize temporal changes. Networks trained on RGB frames may heavily rely on spatial features, which are less informative in event frames. Secondly, RGB images are 2D arrays with three color channels, while event frames represented as histograms are sparse and emphasize different characteristics. The preprocessing steps required for event data, such as creating histograms, can lead to a mismatch in the input format expected by a network trained on RGB data. Moreover, networks pre-trained on RGB frames may have learned features specific to color and intensity patterns, such as edges, textures, and colors. These features are not directly relevant to event data, which needs color information and relies more on motion dynamics. Furthermore, the type and characteristics of noise in RGB frames (e.g., sensor noise, lighting conditions) differ from those in event frames (e.g., event rate variability, sparse data). The networks trained on RGB frames might not be robust to the noise characteristics of event frames. In addition, the architecture of networks designed for RGB frames might need to be optimized for the characteristics of event data. For example, layers and filters tuned for color and spatial features may not effectively capture the temporal dynamics emphasized in event frames. It should be noted, that although the event camera poses remarkable capabilities in terms of low temporal resolution and high dynamic range. The noise associated with the event camera can be a hindering factor in terms of the accuracy and robustness of the methods developed for any task. For example, In low-light conditions, events corresponding to the features or edges of moving objects are widely scattered, and substantial noise remains prevalent despite using optimal camera settings[41]. However, one of the prominent research directions for the event camera is denoising, and many solutions have been proposed that would alleviate this problem and improve the signal to noise ratio of the sensor[42]. Furthermore, event data is less in features compared to RGB cameras since it only perceives edge information, thus event frames tend to be more difficult to learn from compared to conventional sensors.

Code availability

All the codes used for automatic annotation, along with the data needed to use the raw files, are provided in https://github.com/oussamaabdulhay/E_Pose.

Received: 15 July 2024; Accepted: 28 January 2025;
Published online: 12 February 2025

References

1. Park, K.-B., Choi, S. H. & Lee, J. Y. Self-training based augmented reality for robust 3D object registration and task assistance. *Expert Systems with Applications* **238**, 122331 (2024).
2. Vu, V.-D. *et al*. Occlusion-robust pallet pose estimation for warehouse automation. *IEEE Access* (2024).
3. Eppner, C. *et al*. *Lessons from the amazon picking challenge: Four aspects of building robotic systems*. In *Robotics: science and systems*, volume 12 (2016).
4. Hu, Y., Fua, P., Wang, W. & Salzmann, M. Single-Stage 6D Object Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2024).
5. Xiang, Y., Schmidt, T., Narayanan, V. & Fox, D. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *Robotics: Science and Systems XIV* (2018).
6. Saadi, L., Besbes, B., Kramm, S. & Bensrhair, A. Optimizing rgb-d fusion for accurate 6 dof pose estimation. *IEEE Robotics and Automation Letters* **6**(2), 2413–2420 (2021).
7. Hinterstoisser, S. *et al*. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Computer Vision–ACCV 2012*, pages 548–562 (Springer, 2013).
8. Brachmann, E. *et al*. Learning 6d object pose estimation using 3 d object coordinates. In *Computer Vision–ECCV 2014*, pages 536–551 (Springer, 2014).
9. Hodan, T. *et al*. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888 (IEEE, 2017).
10. Calli, B. *et al*. Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set. In *IEEE Robotics Automation Magazine*, **22**(3), 36–52 (2015).
11. Park, K. Patten, T. Prankl, J. & Vincze, M. Multi-Task Template Matching for Object Detection, Segmentation and Pose Estimation Using Depth Images. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7207–7213 (IEEE, 2019).
12. Dubeau, E., Garon, M., Debaque, B., de Charette, R. & Lalonde, J.-F. RGB-DE: Event camera calibration for fast 6-dof object tracking. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–135 (IEEE, 2020).
13. Gallego, G. *et al*. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44**(1), 154–180 (2020).
14. Wang, X. *et al*. Event stream-based visual object tracking: A high-resolution benchmark dataset and a novel baseline. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19248–19257 (2024).
15. Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T. & Scaramuzza, D. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *The International Journal of Robotics Research* **36**(2), 142–149 (2017).
16. Schuman, C. D. *et al*. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science* **2**(1), 10–19 (2022).
17. Furber, S. B., Galluppi, F., Temple, S. & Plana, L. A. The spinnaker project. *Proceedings of the IEEE* **102**(5), 652–665 (2014).
18. Salah, M. *et al*. High speed neuromorphic vision-based inspection of countersinks in automated manufacturing processes. *Journal of Intelligent Manufacturing*, pages 1–15 (2023).
19. Muthusamy, R., Huang, X., Zweiri, Y., Seneviratne, L. & Gan, D. Neuromorphic event-based slip detection and suppression in robotic grasping and manipulation. *IEEE Access* **8**, 153364–153384 (2020).
20. Rigi, A., Naeini, F. B., Makris, D. & Zweiri, Y. A novel event-based incipient slip detection using dynamic active-pixel vision sensor (DAVIS). *Sensors* **18**(2), 333 (2018).

21. Huang, X. *et al.* Real-time grasping strategies using event camera. *Journal of Intelligent Manufacturing* **33**(2), 593–615 (2022).
22. Muthusamy, R. *et al.* Neuromorphic eye-in-hand visual servoing. *IEEE Access* **9**, 55853–55870 (2021).
23. Cao, H., Chen, G., Li, Z., Hu, Y. & Knoll, A. NeuroGrasp: Multimodal Neural Network With Euler Region Regression for Neuromorphic Vision-Based Grasp Pose Estimation. *IEEE Transactions on Instrumentation and Measurement* **71**, 1–11 (2022).
24. Li, B. *et al.* Event-based robotic grasping detection with neuromorphic vision sensor and event-grasping dataset. *Frontiers in neurorobotics* **14**, 51 (2020).
25. Huang, X. *et al.* A neuromorphic dataset for tabletop object segmentation in indoor cluttered environment. *Scientific data* **11**(1), 127 (2024).
26. Rojtberg P. & Pöllabauer, T. YCB-Ev: Event-vision dataset for 6DoF object pose estimation. *arXiv preprint arXiv:2309.08482* (2023).
27. Tekin, B., Sinha, S. N. & Fua. P. Real-time seamless single shot 6 d object pose prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 292–301 (2018).
28. Cao, T. *et al.* DGECN: A depth-guided edge convolutional network for end-to-end 6D pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3783–3792 (2022).
29. Hodan, T. *et al.* BOP: Benchmark for 6D object pose estimation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34 (2018).
30. Liu, H. *et al.* Seeing Motion at Nighttime with an Event Camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 25648–25658 (June 2024).
31. Akinola, I., Xu, J., Song, S. & Allen, P. K. Dynamic grasping with reachability and motion awareness. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9422–9429 (IEEE, 2021).
32. Dornaika, F. & Horaud, R. Simultaneous robot-world and hand-eye calibration. *IEEE Transactions on Robotics and Automation* **14**(4), 617–622 (1998).
33. Li, W., Dong, Y., Qiu, S. & Han, B. Hardware-Free Event Cameras Temporal Synchronization Based on Event Density Alignment. In *International Conference on Intelligent Robotics and Applications*, pages 60–71 (Springer, 2023).
34. Park, J., Zhou, Q.-Y. & Koltun, V. Colored point cloud registration revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 143–152 (2017).
35. Chen, Y. & Medioni, G. Object modelling by registration of multiple range images. *Image and Vision Computing* **10**(3), 145–155 (1992).
36. Hay, A. *et al.* E-POSE: A Large Scale Event Camera Dataset for Object Pose Estimation. *Figshare*, https://doi.org/10.6084/m9.figshare.2810671710.6084/m9.figshare.28106717 (July, 2024).
37. Benosman, R., Ieng, S.-H., Clercq, C., Bartolozzi, C. & Srinivasan, M. Asynchronous frameless event-based optical flow. *Neural Networks* **27**, 32–37 (2012).
38. Zhu, A. Z. & Yuan, L. EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras. In *Robotics: Science and Systems* (2018).
39. Nguyen, A., Do, T.-T., Caldwell, D. G. & Tsagarakis, N. G. Real-time 6DOF pose relocalization for event cameras with stacked spatial LSTM networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0 (2019).
40. Guan, J., Hao, Y., Wu, Q., Li, S. & Fang, Y. A Survey of 6DoF Object Pose Estimation Methods for Different Application Scenarios. *Sensors* **24**(4), 1076 (2024).
41. Alkendi, Y. *et al.* Neuromorphic camera denoising using graph neural network-driven transformers. *IEEE Transactions on Neural Networks and Learning Systems* **35**(3), 4110–4124 (2022).
42. Baldwin, R., Almatrafi, M. Asari, V. & Hirakawa, K. Event probability mask (epm) and event denoising convolutional neural network (edncnn) for neuromorphic cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1701–1710 (2020).
43. Drost, B., Ulrich, M., Bergmann, P., Hartinger, P. & Steger, C. *Introducing MVTec ITODD - A Dataset for 3D Object Recognition in Industry*. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops* (October 2017).
44. Tyree, S. *et al.* 6-DoF Pose Estimation of Household Objects for Robotic Manipulation: An Accessible Dataset and Benchmark. In *International Conference on Intelligent Robots and Systems (IROS)* (2022).
45. Wen, B., Mitash, C., Ren, B. & Bekris, K. E. se(3)-TrackNet: Data-driven 6D Pose Tracking by Calibrating Image Residuals in Synthetic Domains. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct https://doi.org/10.1109/IROS45743.2020.9341314 (2020).

Acknowledgements

This work was performed at the Advanced Research and Innovation Center (ARIC), which is funded by STRATA Manufacturing PJSC (a Mubadala company), Sandooq Al Watan under Grant SWARD-S22-015, and Khalifa University of Science and Technology.

Author contributions

O.H., A.A. and E.S. built the experimental setup; O.H. conducted the data acquisition for all scenarios; O.H. and R.A. completed the 3D reconstruction; O.H. wrote the Matlab code for automatic labeling of events; O.H. and X.H. evaluated the data of this dataset; O.H. organized the dataset; Y.Z., Y.A., L.S. supervised this work and provided revision and editing; A.B and Y.Z. Conducted the project management; Y.Z. is the project principal investigator; all authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Y.Z.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.