

A Framework for Evaluating 6-DOF Object Trackers

Mathieu Garon^[0000–0003–1811–4156], Denis Laurendeau^[0000–0003–2858–5955] and
Jean-François Lalonde^[0000–0002–6583–2364]

Université Laval**

Abstract. We present a challenging and realistic novel dataset for evaluating 6-DOF object tracking algorithms. Existing datasets show serious limitations—notably, unrealistic synthetic data, or real data with large fiducial markers—preventing the community from obtaining an accurate picture of the state-of-the-art. Using a data acquisition pipeline based on a commercial motion capture system for acquiring accurate ground truth poses of real objects with respect to a Kinect V2 camera, we build a dataset which contains a total of 297 calibrated sequences. They are acquired in three different scenarios to evaluate the performance of trackers: *stability*, robustness to *occlusion* and accuracy during challenging *interactions* between a person and the object. We conduct an extensive study of a deep 6-DOF tracking architecture and determine a set of optimal parameters. We enhance the architecture and the training methodology to train a 6-DOF tracker that can robustly generalize to objects never seen during training, and demonstrate favorable performance compared to previous approaches trained specifically on the objects to track.

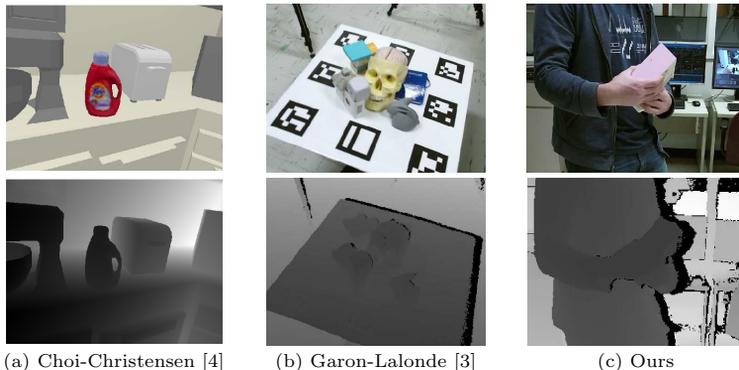
Keywords: 3D object tracking, databases, deep learning

1 Introduction

With the recent emergence of 3D-enabled augmented reality devices, tracking 3D objects in 6 degrees of freedom (DOF) is a problem that has received increased attention in the past few years. As opposed to SLAM-based camera localization techniques—now robustly implemented on-board various commercial devices—that can use features from the entire scene, 6-DOF object tracking approaches have to rely on features present on a (typically small) object, making it a challenging problem. Despite this, recent approaches have demonstrated tremendous performance both in terms of speed and accuracy [1–3].

Unfortunately, obtaining an accurate assessment of the performance of 6-DOF object tracking approaches is becoming increasingly difficult since accuracy on the main dataset used for this purpose has now reached closed to 100%. Introduced in 2013 by Choi and Christensen [4], their dataset consists of 4 short sequences of purely synthetic scenes. The scenes are made of unrealistic, textureless backgrounds with a single colored object to track, resulting in noiseless

** mathieu.garon.2@ulaval.ca, denis.laurendeau@gel.ulaval.ca, jflalonde@gel.ulaval.ca



(a) Choi-Christensen [4]

(b) Garon-Lalonde [3]

(c) Ours

Fig. 1. Comparison of datasets for evaluating 6-DOF tracking algorithms. Typical RGB (top) and depth (bottom) frames for (a) the synthetic dataset of Choi and Christensen [4], (b) the real dataset of Garon and Lalonde [3], and (c) ours. Compared to the previous work, our dataset contains real objects captured by a sensor, and does not use a calibration board, therefore mimicking realistic real-world scenarios.

RGBD images (see fig. 1-(a)). The object is static and the camera rotates around it in wide motions, occasionally creating small occlusions (at most 20% of the object is occluded). While challenging at first, the dataset has now essentially been solved for the RGBD case. For example, the method of Kehl et al. [1] (2017) reports an average error in translation/rotation of 0.5mm/0.26°, which is an improvement of 0.3mm/0.1° over the work of Tan et al. (2015) [5], who have themselves reported a 0.01mm/1° improvement to the approach designed by Krull et al. (2014) [6]. The state of the art on the dataset has reached a near-perfect error of 0.1mm/0.07° [2], which highlights the need for a new dataset with more challenging scenarios.

Another dataset, introduced by Garon and Lalonde [3], includes 12 sequences of real objects captured with real sensors. While a significant improvement over the synthetic dataset of [4], dealing with real data raises the issue of providing accurate ground truth pose of the object at all times. To obtain this ground truth information, their strategy (also adopted in 6-DOF *detection* datasets [7, 8]) is to use calibration boards with fiducial markers. While useful to accurately and easily determine an object pose, this has the unfortunate consequence of constraining the object to lie on a large planar surface (fig. 1-(b)).

In this paper, we present a novel dataset allowing the systematic evaluation of 6-DOF tracking algorithms in a wide variety of real scenarios without requiring calibration boards (fig. 1-(c)). Our dataset is one order of magnitude larger than the previous work: it contains 297 sequences of 11 real objects. The sequences are split into 3 different scenarios, which we refer to as *stability*, *occlusion*, and *interaction*. The *stability* scenario aims at quantifying the degree of jitter in a tracker. The object is kept static and placed at various angles and distances from the camera. The *occlusion* scenario, inspired by [3], has the object rotating on a turntable and being progressively occluded by a flat panel. Occlusion ranges from

0% (unoccluded) to 75%, thereby testing trackers in very challenging situations. Finally, in the *interaction* scenario, a person is moving the object around freely in front of the camera (fig. 1-(c)), creating occlusions and varying object speed.

In addition, we also introduce two new 6-DOF real-time object trackers based on deep learning. The first, trained for a specific object, achieves state-of-the-art performance on the new dataset. The second, trained *without a priori knowledge of the object to track*, is able to achieve an accuracy that is comparable with previous work trained specifically on the object. These two trackers rely on the same deep learning architecture and only differ in the training data. Furthermore, both of our trackers have the additional significant advantage of requiring only *synthetic* training data (i.e. no real data is needed for training). We believe this is an exciting first step in the direction of training generic trackers which do not require knowledge of the object to track at training time.

In summary, this paper brings 3 key contributions to 6-DOF object tracking:

1. A novel dataset of real RGBD sequences for the systematic evaluation of 6-DOF tracking algorithms that is one order of magnitude larger than existing ones, and contains 3 challenging scenarios;
2. A real-time deep learning architecture for tracking objects in 6-DOF which is more stable and more robust to occlusions than previous approaches;
3. A generic 6-DOF object tracker trained without knowledge of the object to track, achieving performance on par with previous approaches trained specifically on the object.

2 Related work

There are two main relevant aspects in 6-DOF pose estimation: single frame object detection and multi-frame temporal trackers. The former has received a lot of attention in the literature and benefits from a large range of public datasets. The most notorious dataset is arguably *Linemod* [7], which provide 15 objects with their mesh models and surface colors. To obtain the ground truth object pose, a calibration board with fiducial markers is used. Since then, many authors created similar but more challenging benchmarks [8–10]. However these datasets do not contain temporal and displacement correlation between each frame, which makes them inadequate for evaluating temporal trackers.

In the case of temporal tracking, only a few datasets exist to evaluate the approaches. As mentioned in the introduction, the current, widely used standard dataset is the synthetic dataset of Choi and Christensen [4], which contains 4 sequences with 4 objects rendered in a texture-less virtual scene. Another available option is the one provided by Akkaladevi et al. [11] who captured a single sequence of a scene containing 4 different objects with a Primesense sensor. However, the 3D models are not complete and do not include training data that could be exploited by learning-based methods. Finally, recent work by Garon and Lalonde [3] proposed a public dataset of 4 objects containing 4 sequences with clutter and an additional set of 8 sequences with controlled occlusion on a specific object. Fiducial markers are used to generate the ground truth pose

of the model, which limits the range of displacements that can be achieved. In contrast, we propose a new method to collect ground truth pose data that makes the acquisition simpler without the need for fiducial markers.

There is an increasing interest in 6-DOF temporal trackers since they were shown to be faster and more robust than single frame detection methods. In the past, geometric methods based on ICP [4,12–14] were used for temporal tracking, but they lack robustness for small objects and are generally computationally expensive. Data-driven approaches such as the ones reported in [5,6,15] can learn more robust features and the use of the Random Forest regressor [16] decreases the computing overhead significantly. Other methods show that the contours of the objects in RGB and depth data provide important cues for estimating pose [1, 2,17]. While their optimization techniques can be accurate, many assumptions are made on the features which restrict the type of object and the type of background that can be dealt with. Recently, Garon and Lalonde [3] proposed a deep learning framework which can learn robust features automatically from data. They use a feedback loop by rendering the 3D model at runtime at the previous pose, and regress the pose difference between the rendered object and the real image. While their method compares to the previous work with respect to accuracy, their learned features are more robust to higher level of occlusion and noise. A downside is that their method needs a dataset of real images and a specific network has to be trained for each object which can be time consuming. We take advantage of their architecture but introduce novel ideas to provide a better performing tracker that can be trained entirely on synthetic data. In addition, our network can be trained to generalize to previously unseen objects.

3 Dataset capture and calibration

Building a dataset with calibrated object pose w.r.t the sensor at each frame is a challenging task since it requires an accurate method to collect ground truth object pose. Until now, the most practical method to achieve this task was to use fiducial markers and calibrate the object pose w.r.t these markers [3,7–9]. However, this method suffers from two main drawbacks. First, the object cannot be moved independently of the panel so this restricts the camera to move around the object of interest. Second, the scene always contains visual cues (the markers) which could involuntarily “help” the algorithms.

Our approach eliminates these limitations. A ViconTM MX-T40 motion capture system is used to collect the ground truth pose of the objects in the scene. The retroreflective Vicon markers that must be used are very small in size (3mm diameter) and can be automatically removed in a post-processing step. In this section, we describe the capture setup and the various calibration steps needed to align the object model and estimate its ground truth pose. The resulting RGBD video sequences captured using this setup are presented in sec. 4.

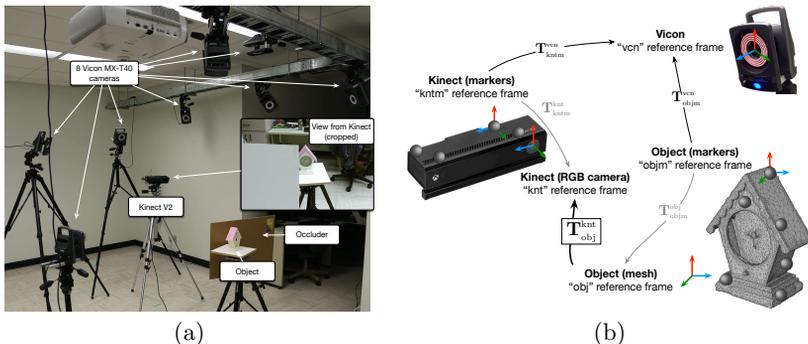


Fig. 2. Acquisition setup used to capture our novel dataset. (a) Actual setup, which includes an 8-camera Vicon motion capture system and a Kinect V2. The resulting view from the Kinect is shown in the inset. Here, an occluder is placed in front of the object. (b) The various transformations that must be calibrated in order to obtain the object pose in the Kinect RGB camera reference frame \mathbf{T}_{obj}^{knt} . The transformations shown in black are obtained from the motion capture system directly, while the gray ones need a specific calibration procedure described in the main body of the paper.

3.1 Capture setup

The motion capture setup is composed of a set of 8 calibrated cameras that track retro-reflective markers of 3mm in diameter installed on the objects of interest in a $3 \times 3 \times 3\text{m}^3$ work area. Vicon systems can provide a marker detection accuracy of up to 0.15 mm on static objects and 2mm on moving objects according to [18]. A Kinect V2 is used to acquire the RGBD frames, and is calibrated with the Vicon to record the ground truth pose of the objects in the Kinect coordinate system. The actual setup used to capture the dataset is shown in fig. 2-(a).

3.2 Calibration

With an RGB-D sensor such as the Kinect V2, color and depth values are projected onto two different planes. We define the Kinect reference frame (“knt”) as the origin of its RGB camera, and align the depth data by reprojecting it to the color plane using the factory calibration parameters. We calibrate the depth correction as in Hodan et al. [8]. In this section, the notation \mathbf{T}_a^b is used to denote a rigid transformation from reference frame “a” to “b”.

We aim to recover the pose of the object in the Kinect reference frame \mathbf{T}_{obj}^{knt} (fig. 2-(b)). To do so, we first rely on the Vicon motion capture system, which has its own reference frame “vcn”. The set of retroreflective markers installed on the object define the local reference frame “objm”. Similarly, the set of markers placed on the Kinect define the local reference frame “kntm”. The Vicon system provides the transformations \mathbf{T}_{objm}^{vcn} and \mathbf{T}_{kntm}^{vcn} directly, that is, the mappings between the object and Kinect markers and the Vicon reference frame respectively. The transformation between the object markers and the Kinect markers

is obtained by chaining the previous transformations:

$$\mathbf{T}_{objm}^{kntm} = (\mathbf{T}_{kntm}^{vcn})^{-1} \mathbf{T}_{objm}^{vcn} . \quad (1)$$

The pose \mathbf{T}_{obj}^{knt} is recovered with the transformations between the local frames defined by the markers and the object/Kinect reference frames $\mathbf{T}_{objm}^{objjm}$ and \mathbf{T}_{kntm}^{kntm} :

$$\mathbf{T}_{obj}^{knt} = (\mathbf{T}_{kntm}^{kntm})^{-1} \mathbf{T}_{objm}^{kntm} \mathbf{T}_{objm}^{objjm} . \quad (2)$$

The calibration procedures needed to obtain these two transformations, also shown in gray in fig. 2-(b), are detailed next.

Kinect calibration In order to find the transformation \mathbf{T}_{kntm}^{knt} between the local frame defined by the markers installed on the Kinect and its RGB camera, we rely on a planar checkerboard target on which Vicon markers are randomly placed. Then, the position of each corner of the checkerboard is determined with respect to the markers with the following procedure. A 15cm-long pen-like probe that has a 1cm Vicon marker attached at one end was designed for this purpose. The sharp end is placed on the corner to be detected, and the probe is moved in a circular motion around that point. A sphere is then fit (using least-squares) to the resulting marker positions (achieving an average radius estimation error of 0.7mm), and the center of the sphere is kept as the location for the checkerboard corner. The checkerboard target was then moved in the capture volume and corners were detected by the Kinect RGB camera, thereby establishing 2D-3D correspondences between these points. The perspective- n -points algorithm [19] was finally used to compute \mathbf{T}_{kntm}^{knt} .

Object calibration To estimate the transformation between the local frame defined by the markers placed on the object and its mesh coordinate system \mathbf{T}_{objjm}^{obj} , we rely on the Kinect pose calibrated with the method described previously. As a convention, we define the origin of the object local coordinate system at the center of mass of the markers, the same convention is used for the mesh by using the center of mass of the vertices. We roughly align the axis and use ICP to refine its position (based on the Kinect depth values). Finally, with the help of a visual interface where a user can move and visualize the alignment of the object, fine-scale adjustments can be performed manually from several viewpoints to minimize the error between the observed object and the reprojected mesh.

Synchronization In addition to spatial calibration, precise temporal alignment must be achieved to synchronize Vicon and Kinect frames. Unfortunately, the Kinect does not offer hardware synchronization capabilities, therefore we adopt the following software solution. We assume that the sequences are short enough to neglect clock drift. We also assume a stable sampling of the Vicon system on a high bandwidth closed network. In this setup, synchronization can be achieved by estimating the (constant) time difference δt between the Vicon and the Kinect

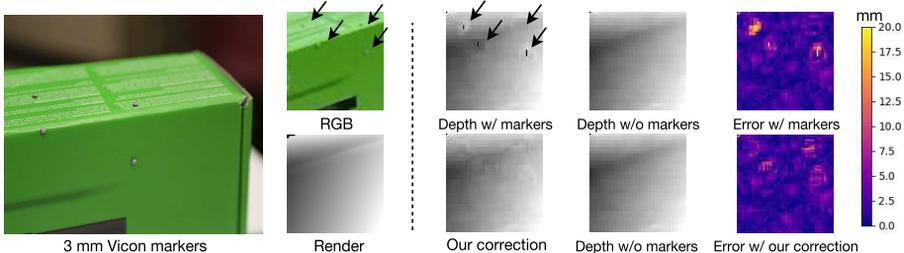


Fig. 3. Example of an RGB and depth frame containing 2 markers on a flat surface, and 2 markers near an edge. We take advantage of our knowledge of the object mesh and pose to replace patches of 10×10 pixels around the marker by the depth values of a render at the same pose. We capture an image without the markers to compare the error. On the modified patches we report a RMSE of 139.8 mm on the depth with the markers, and 4.7 with the corrected version.

frame timestamps. By moving the checkboard of sec.3.2 with varying speed, we estimate the δt that minimizes the reprojection error between the checkerboard corners from sec. 3.2 and the Vicon markers.

Removing the markers The 3mm markers used to track the object are retro-reflective and, despite their small size and their low number (7 per object on average), they nevertheless create visible artifacts in the depth data measured by the Kinect, see fig. 3. We propose a post-processing algorithm for automatically removing them in all the sequences. First, to ensure that the marker can be observed by the Kinect we reproject the (known) marker positions onto the depth image and compute the median distance between the depth in a small window around the reprojected point and its ground truth depth. If the difference is less than 1cm, the point is considered as not occluded, and will be processed. Finally, we render the depth values of the 3D model at the given pose and replace the 10×10 pixel window from the original image with the rendered depth values. For more realism, a small amount of gaussian noise is added. Pixels from the background are simply ignored. On average, only 3.4% of the object pixels are corrected. We also minimize the chances of affecting the geometric structure of the object by placing the markers on planar surfaces. Fig. 3 shows a comparison of the error between a Kinect depth image captured with markers, and another image of the same scene with markers that have been corrected with our algorithm. The RMSE of the pixel patches around the markers is 139.8 mm without the correction, and 4.7 mm with the correction.

4 Dataset scenarios, metrics, and statistics

This section defines novel ways to evaluate 6-DOF trackers using calibrated sequences captured with the setup presented in sec. 3. We provide an evaluation

methodology that will reflect the overall performance of a tracker in different scenarios. To attain this objective, we captured 297 sequences of 11 different objects of various shapes in 3 scenarios: *stability*, *occlusion*, and *interaction*. We also provide quantitative metrics to measure the performance in each scenario. Our dataset and accompanying code is available at <http://www.jflalonde.ca/projects/6dofObjectTracking>.

4.1 Performance metrics

Before we describe each scenario, we first introduce how we propose to evaluate the difference between two poses \mathbf{P}_1 and \mathbf{P}_2 . Here, a pose $\mathbf{P} = [\mathbf{R} \ \mathbf{t}]$ is described by a rotation matrix \mathbf{R} and a translation vector \mathbf{t} . Previous works consider the average of each axis component in translation and rotation separately. The side effect of this metric is that a large error on a single component is less penalized. To overcome this limitation, the translation error is simply defined as the L2 norm between the two translation vectors:

$$\delta_{\mathbf{t}}(\mathbf{t}_1, \mathbf{t}_2) = \|\mathbf{t}_1 - \mathbf{t}_2\|_2. \quad (3)$$

The distance between two rotation matrices is computed using:

$$\delta_{\mathbf{R}}(\mathbf{R}_1, \mathbf{R}_2) = \arccos\left(\frac{\text{Tr}(\mathbf{R}_1^T \mathbf{R}_2) - 1}{2}\right), \quad (4)$$

where $\text{Tr}(\cdot)$ denotes the matrix trace.

4.2 Scenarios

The *stability* scenario In this first scenario, we propose to quantify the degree of pose jitter when tracking a static object. To evaluate this, we captured 5-second sequences of the object under 4 different viewpoints and with 3 configurations: at a distance of 0.8m from the sensor (“near”), of 1.5m from the sensor (“far”), and of 0.8m from the sensor, but this time with distractor objects partly occluding the object of interest (“occluded”). To measure the stability, Tan et al. [2] use the standard deviation of the pose parameters on a sequence. We propose a different metric inspired from [20] that penalizes variation from frame to frame instead of the general distribution across the sequence. We compute the distance between poses \mathbf{P}_{i-1} and \mathbf{P}_i at time i . In other words, we report the distribution of $\delta_{\mathbf{t}}(\mathbf{t}_{i-1}, \mathbf{t}_i)$ and $\delta_{\mathbf{R}}(\mathbf{R}_{i-1}, \mathbf{R}_i)$ for all frames of the stability scenario.

The *occlusion* scenario To evaluate the robustness to occlusion, we follow [3] and place the object on a turntable at 1.2m from the sensor, and a static occluder is placed in front of the object in a vertical and horizontal position. We compute the amount of occlusion based on the largest dimension of the object, and provide sequences for each object from 0% to 75% occlusion in 15%



Fig. 4. Overview of the 11 objects in our dataset, with their maximum distance between two vertices in mm shown above.

increments, which results in a total of 11 sequences per object. Here, we compute errors by comparing the pose \mathbf{P}_i at time i with the ground truth \mathbf{P}_i^* for that same frame, i.e., $\delta_{\mathbf{t}}(\mathbf{t}_i^*, \mathbf{t}_i)$ and $\delta_{\mathbf{R}}(\mathbf{R}_i^*, \mathbf{R}_i)$. Temporal trackers may lose tracking on difficult frames. This can affect the overall score depending on the moment where the tracker fails. To bypass this limitation, we initialize the tracker at the ground truth pose \mathbf{P}_i^* every 15 frames as in [3].

The *interaction* scenario In this last scenario, the experimenter holds the object in his hands and manipulates it in 4 different ways: 1) by moving the object around but without rotating it (“translation-only”); 2) by rotating the object on itself without translating it (“rotation-only”); 3) by freely moving and rotating the object around at low speeds (“free-slow”); and 4) by freely moving and rotating the object at higher speeds and by voluntarily generating more occlusions (“free-hard”). In all situations but the “free-hard”, we reset the tracker every 15 frames and we report $\delta_{\mathbf{t}}(\mathbf{t}_i^*, \mathbf{t}_i)$ and $\delta_{\mathbf{R}}(\mathbf{R}_i^*, \mathbf{R}_i)$ as in sec. 4.2. Since the object speed varies, we also compute the translational and rotational inter-frame displacement ($\delta_{\mathbf{t}}(\mathbf{t}_{i-1}^*, \mathbf{t}_i^*)$, $\delta_{\mathbf{R}}(\mathbf{R}_{i-1}^*, \mathbf{R}_i^*)$) and report the performance metric above as a function of that displacement. In addition, it is also informative to count the number of times the tracker has failed. We consider a tracking failure when either $\delta_{\mathbf{t}}(\mathbf{t}_i^*, \mathbf{t}_i) > 3\text{cm}$ or $\delta_{\mathbf{R}}(\mathbf{R}_i^*, \mathbf{R}_i) > 20^\circ$ for more than 7 consecutive frames. When a failure is detected, the tracker is reset at the ground truth pose \mathbf{P}^* . We report these failures on the “free-hard” sequences only.

4.3 Dataset statistics

We selected 11 different objects to obtain a wide variety of object geometries and appearance, as shown in fig. 4. To obtain a precise 3D model of each object in the database, each of them was scanned with a Creafom GoScanTM handheld 3D scanner at a 1mm voxel resolution. The scans were manually cleaned using Creafom VxElementsTM to remove background and spurious vertices.

Overall, the dataset contains 297 sequences: 27 sequences for each object. The breakdown is the following: 12 sequences for *stability* (4 viewpoints, 3 configurations: “near”, “far”, “occluded”); 11 sequences for *occlusion* (0% to 75%

Fig. 5. The deep learning architecture used to track 3D objects in this work, inspired by [3]. The notation “conv x - y ” indicates a convolution layer of y filters of dimension $x \times x$, “fire- x - y ” indicates a “fire” module [21] which reduces the number of channels to x and expands to y , and “FC- x ” is a fully-connected layer of x units. Each layer have a skip link similar to DenseNet [22] and is followed by a max pooling 2×2 operation. We use a dropout of 50% on the input connections to the FC-500 layer. All layers (except the last FC-6) have batch normalization and the ELU activation function [23].

Input: \mathbf{x}_{pred}	Input: \mathbf{x}_{obs}
conv3-96	conv3-96
fire-48-96	fire-48-96
concatenation	
fire-96-384	
fire-192-768	
fire-384-768	
FC-500	
FC-6	
Output: \mathbf{y}	

in 15% increment for both horizontal and vertical occluders); and 4 sequences for *interaction* (“rotation-only”, “translation-only”, “free-slow”, “free-hard”). It also contains high resolution textured 3D models for each object.

5 Analyzing a deep 6-DOF tracker with our dataset

As a testbed to evaluate the relevance of the new dataset, we borrow the technique of Garon and Lalonde [3] who train a 6-DOF tracker using deep learning, but propose changes to their architecture and training methodology. We evaluate several variants of the network on our dataset and show that it can be used to accurately quantify the performance of a tracker in a wide variety of scenarios.

5.1 Training an object-specific tracker

We propose improvements over the previous work of [3] by adding 5 main changes: 2 to the network architecture, and 3 to the training procedure. The new proposed network architecture is shown in fig. 5. As in [3], the network accepts two inputs: an image of the object rendered at its predicted position (from the previous timestamp in the video sequence) \mathbf{x}_{pred} , and an image of the observed object at the current timestamp \mathbf{x}_{obs} . The last layer outputs the 6-DOF (3 for translation, 3 for rotation in Euler angles) representing the pose change between the two inputs. We first replace convolution layers by the “fire” modules proposed in [21]. The second change, inspired by DenseNet [22], is to concatenate the input features of each layers to the outputs before being max-pooled. Our improvement requires the same runtime as [3], which is 6 ms on a Nvidia GTX-970M. As in [3], the loss used is the MSE between the predicted and ground truth pose change. Note that we experimented with the reprojection loss [24], but found it did not help in our context.

We also propose changes to the training procedure of [3]. Their approach consists in generating pairs of synthetic views of the object with random pose

changes between them. To sample the random pose changes, they proposed to independently sample a random translation $t_{x,y,z} \sim U(20\text{mm}, 20\text{mm})$ and rotation $r_{\alpha,\beta,\gamma} \sim U(-10^\circ, 10^\circ)$ in Euler angle notation, with $U(a, b)$ referring to a uniform distribution on interval $[a, b]$. Doing so unfortunately biases the resulting pose changes. For example, small amplitude translations are quite unlikely to be generated (since this requires *all* three translation components to be small simultaneously). Our first change is to sample a random translation *vector* and *magnitude* separately. The translation vector \mathbf{v}_t is sampled in spherical coordinates (θ_t, ϕ_t) , where $\theta_t \sim U(-180^\circ, 180^\circ)$ and $\phi_t = \cos^{-1}(x)$ with $x \sim U(-1, 1)$. The translation magnitude m_t is drawn from a Gaussian distribution $m_t \sim \mathcal{N}(0, \Delta t)$. The same process is repeated for rotations, where the rotation axis \mathbf{v}_r and angle $m_r \sim \mathcal{N}(0, \Delta r)$ are sampled similarly. Here, we intentionally parameterize the translation magnitude m_t and rotation angle m_r distributions with Δt and Δr , since the range of these parameters may influence the behavior of the network. Our second change is to downsample the depth channel to better match the resolution of the Kinect V2. Our third change consists in a data augmentation method for RGBD images where we randomly set a modality (depth or RGB) to zero during training, which has the effect of untangling the features of both modalities. With these changes, we can now rely purely on synthetic data to train the network (in [3] a set of real frames was required to fine-tune the network).

5.2 Training a generic tracker

To train a generic 6-DOF object tracker, we experimented with two ways of generating a training dataset, using the same network architecture, loss, and training procedure described in sec. 5.1. First, we generate a training set of images that contain all 11 objects from our dataset, as well as 30 other objects. These other objects, downloaded from 3D Warehouse¹ and from “Linemod” [7], show a high diversity in geometry and texture and are roughly of the same size. We name the network trained on this dataset the “multi-object” network. Second, we generate a training set of images that contain only the 30 *other* objects—the actual objects to track are *not* included. We call this network “generic”, since it never saw any of the objects in our dataset during training. Note that all these approaches require the 3D model of the object to track at test time, however.

6 Experiments

In this section, we perform an exhaustive evaluation of the various approaches presented in sec. 5 using our novel dataset and framework proposed in sec. 4. First, we analyze the impact of varying the training data generation hyperparameters Δt and Δr for the object-specific case. Then, we proceed to compare our object-specific, “multi-object”, and “generic” trackers with two existing methods: Garon and Lalonde [3] and Tan et al. [5].

¹ Available at: <https://3dwarehouse.sketchup.com>.

Stability Scenario		Translation (mm/frame)			Rotation (degree/frame)			
		Near	Far	Occluded	Near	Far	Occluded	
Δt	10	0.42	0.53	0.48	15	0.35	0.51	0.63
	20	0.62	0.77	0.71	20	0.55	0.68	0.88
	30	0.72	0.84	0.89	25	0.53	0.66	0.97
	40	0.75	0.89	1.00	30	0.61	0.74	1.04
	50	0.95	1.03	1.16	35	0.68	0.87	1.10

Occlusion scenario		Translation (mm)					Rotation (degrees)							
		0	15	30	45	60	75	0	15	30	45	60	75	
Δt	10	14.8	12.5	13.1	15.5	20.0	25.3	15	2.1	2.9	5.4	8.1	15.7	26.0
	20	7.5	7.6	12.5	15.5	22.3	29.5	20	3.3	4.8	7.5	9.9	17.2	48.0
	30	11.0	11.5	17.4	21.5	26.6	33.9	25	3.3	4.8	8.7	16.8	30.6	41.1
	40	12.7	14.8	17.9	26.0	36.0	68.1	30	2.7	3.9	6.9	12.3	26.9	62.9
	50	10.4	11.1	17.7	30.6	43.8	73.7	35	3.2	4.6	9.1	16.1	36.7	66.1

Interaction scenario		Translation (mm)				Rotation (degrees)				
		(0, 10]	(10, 20]	(20, 30]	(30, 40]	(0, 4]	(4, 8]	(8, 12]	(12, 16]	
Δt	Speed (per frame)									
	10	10.1	11.8	16.4	18.8	15	2.9	3.8	4.2	4.4
	20	6.5	8.7	11.0	18.0	20	3.7	4.7	4.9	5.0
	30	9.7	9.9	11.5	9.5	25	5.7	5.9	5.8	6.1
	40	10.8	11.4	11.7	6.6	30	4.3	4.6	4.7	4.7
50	10.5	11.4	10.9	7.2	35	4.7	4.8	5.4	5.1	

(a) Impact of Δt on δ_t
(b) Impact of Δr on δ_R

Table 1. Applying our evaluation methodology for determining the best range of translations Δt and rotations Δr for generating synthetic data when training a deep 6-DOF tracker. We show (a) the impact of varying Δt on the error δ_t , and (b) the impact of varying Δr on the error δ_R for all three scenarios (from top to bottom: *stability*, *occlusion*, and *interaction*).

6.1 Analysis to training data generation parameters

We now apply the evaluation methodology proposed in sec. 4 on the method presented above and evaluate the influence of the Δr and Δt hyper-parameters on the various metrics and sequences from our dataset. We experiment by varying $\Delta t \in \{10, 20, 30, 40, 50\}$ mm and $\Delta r \in \{15, 20, 25, 30, 35\}$ ° one at a time (the other parameter is kept at its lowest value). For each of these parameters, we synthesize 200,000 training image pairs per object using [3] and the modifications proposed in sec. 5.1. We then train a network for each object, for each set of parameters, and evaluate each network on our dataset. A subset of the results of this analysis is shown in tab. 1. Note that, for the *interaction* scenario, the “free-hard” sequences (sec. 4.2) were left out since they are much harder than the others and would bias the results. In particular, we show the impact that varying Δt has on δ_t , as well as that of varying Δr has on δ_R for all 3 scenarios. Here, we drop the parentheses for the $\delta_{\{t, R\}}$ error metrics for ease of notation (see sec. 4 for the definitions). The figure reveals a clear trend: increasing Δr (tab. 1-(b)) systematically results in worse performance in rotation. This is especially visible for the high occlusion cases (45% and 60%), where the rotation error δ_R increases significantly as a function of Δr . The situation is not so simple when Δt is increased (tab. 1-(a)). Indeed, while increasing Δt negatively impacts δ_t in the *stability* and *occlusion* scenarios, performance actually *improves* when the object speed is higher, as seen in the *interaction* scenario. Therefore, to achieve a good balance between stability and accuracy at higher speeds, a value of $\Delta t = 30$ mm seems to be a good trade-off. The remainder of the plots for this analysis, as

Stability scenario	Translation (mm/frame)			Rotation (degree/frame)		
	Near	Far	Occluded	Near	Far	Occluded
Ours specific	0.56	0.68	0.72	0.52	0.59	0.76
Ours multi-object	0.38	0.41	0.57	0.69	0.79	1.09
Ours generic	0.72	0.75	1.19	0.95	0.98	1.67
Garon and Lalonde [1]	0.93	1.06	1.24	1.13	1.23	1.49
Tan et al [5]	1.20	1.31	1.53	1.30	1.44	1.92

Occlusion scenario	Translation (mm)					Rotation (degrees)						
	0	15	30	45	75	0	15	30	45	75		
Ours specific	7.4	9.8	11.5	12.5	15.5	24.2	3.6	5.9	7.9	10.0	12.6	22.2
Ours multi-object	23.3	16.9	14.6	14.3	13.2	13.4	4.0	8.6	12.3	12.1	14.7	15.4
Ours generic	6.7	11.1	18.9	25.9	34.4	47.4	5.3	8.4	16.1	26.8	43.7	50.6
Garon and Lalonde [1]	7.4	11.2	18.9	26.8	38.1	55.0	5.3	8.8	17.7	28.2	41.7	49.8
Tan et al [5]	8.2	8.5	15.9	138	186	213	4.0	7.4	33.1	70.3	89.5	88.0

Interaction scenario	Translation (mm)				Rotation (degrees)				Fail
	(0, 12.5]	(12.5, 25]	(25, 37.5]	(37.5, 50]	(0, 19]	(19, 37]	(37, 56]	(56, 75]	
Ours specific	8.2	10.3	11.1	13.4	3.7	5.8	3.6	5.8	37
Ours multi-object	22.1	27.3	26.0	41.9	6.0	8.6	2.9	6.1	127
Ours generic	9.3	9.9	11.7	13.4	6.3	6.8	8.8	7.0	38
Garon and Lalonde [1]	9.5	10.2	10.3	12.4	7.8	9.5	13.4	11.8	53
Tan et al [5]	8.1	8.5	10.7	67.1	4.5	6.0	8.1	10.1	86

Table 2. Comparison of our networks with the previous work of [3] and [5]. Our “object-specific” networks outperform the state of the art in almost all scenarios, and performs remarkably well at predicting the rotation. Our “generic” tracker shows great promise: although not as good as the “object-specific” version, it results in slightly lower error compared with [3], even if it has not seen any of these objects during training. See the supplementary video for a visual qualitative comparison of the trackers.

well as plots evaluating the impact of the resolution of the crop and the size of the bounding box w.r.t the object are shown in the supplementary material.

6.2 Comparison with previous work

Our trackers yields a 1.7mm/0.6° error on the 4 sequences of [4] which is slightly above [5] who obtain 0.81mm/0.37°. However, as reported in tab. 2, more interesting differences between the trackers can be observed when using our new dataset. We compare with object-specific versions of the work of Garon and Lalonde [3] as well as the Random Forest approach of Tan et al. [5]. For [3], we use the training parameters reported in their paper. For our trackers, the Δm and Δr hyper-parameters were obtained with leave-one-out cross-validation to ensure no training/test overlap. As before, the “free-hard” sequences were left out for the *interaction* experiments.

Overall, as can be observed in tab. 2, the proposed deep learning methods perform either on par or better than the previous work. The “object-specific” networks outperform almost all the other techniques, except for the case of translational error in the interaction scenario. It performs remarkably well at predicting rotations, and is on par with the other methods for translation. In comparison, [5] performs well at low occlusions, but fails when the occlusion level is 30% or greater (particularly in rotation). [3] shows improved robustness to occlusions, but still achieves high rotation errors at 45% occlusion, and is also much less stable (esp. in rotation) than our “object-specific” networks. Interestingly, our “generic” tracker, which has seen none of these objects in training,

performs similarly to the previous works that were trained specifically on these objects. Indeed, it shows a stability, robustness to occlusions and behavior at higher speeds that is similar to [3] and [5], demonstrating that learning generic features that are useful for tracking objects can be achieved. Finally, we use the “free-hard” *interaction* sequences to count the number of times the tracking is lost (sec. 4.2). In this case, the “object-specific” and “generic” networks outperforms the other methods. Qualitative videos showing side-by-side comparisons of these methods are available in the supplementary material.

7 Discussion

The recent evolution in 6-DOF tracking performance on the popular dataset of Choi et al. [4] highlights the need for a new dataset containing real data and more challenging scenarios. In this paper, we provide such a dataset, which we hope will spur further research in the field. Our dataset contains 297 sequences containing 11 objects of various shapes and textures. The sequences are grouped into 3 scenarios: *stability*, *occlusion*, and *interaction*. The dataset and companion evaluation code is released publicly². Additionally, we build on the framework of [3] with an improved architecture and training procedure which allows the network to learn purely from synthetic data, yet generalize well on real data. In addition, the architecture allows for training on multiple objects and test on *different* objects it has never seen in training. To the best of our knowledge, we are the first to propose such a generic learner for the 6-DOF object tracking task. Finally, our approach is extensively compared with recent work and is shown to achieve better performance.

A current limitation is that the Vicon markers must be removed in a post-processing step, which may leave some artifacts behind. While the markers are very small (3mm) and the resulting marker-free images have low error (see fig. 3), there might still be room for improvement. Finally, our “generic” tracker is promising, but it still does not perform quite as well as “object-specific” models, especially for rotations. In addition, a 3D model of the object is still required at test time, so exploring how this constraint can be removed would make for an exciting future research direction.

Acknowledgements

The authors wish to thank Jonathan Gilbert for his help with data acquisition and Sylvain Comtois for the Vicon setup. This work was supported by the NSERC/Creaform Industrial Research Chair on 3D Scanning: CREATION 3D. We gratefully acknowledge the support of Nvidia with the donation of the Tesla K40 and Titan X GPUs used for this research.

² <http://www.jflalonde.ca/projects/6dof0bjectTracking>

References

1. Kehl, W., Tombari, F., Ilic, S., Navab, N.: Real-time 3D model tracking in color and depth on a single CPU core. In: IEEE Conference on Computer Vision and Pattern Recognition. (2017)
2. Tan, D.J., Navab, N., Tombari, F.: Looking beyond the simple scenarios: Combining learners and optimizers in 3D temporal tracking. *IEEE transactions on visualization and computer graphics* **23**(11) (2017) 2399–2409
3. Garon, M., Lalonde, J.F.: Deep 6-DOF tracking. *IEEE Transactions on Computer Graphics and Visualization* **23**(11) (2017)
4. Choi, C., Christensen, H.I.: RGB-D object tracking: A particle filter approach on GPU. In: International Conference on Intelligent Robots and Systems. (2013)
5. Tan, D.J., Tombari, F., Ilic, S., Navab, N.: A versatile learning-based 3D temporal tracker: Scalable, robust, online. In: IEEE International Conference on Computer Vision. (2015)
6. Krull, A., Michel, F., Brachmann, E., Gumhold, S., Ihrke, S., Rother, C.: 6-dof model based tracking via object coordinate regression. In: Asian Conference on Computer Vision. (2014)
7. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: Asian conference on computer vision. (2012)
8. Hodan, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., Zabulis, X.: T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. In: IEEE Winter Conference on Applications of Computer Vision. (2017)
9. Tejani, A., Tang, D., Kouskouridas, R., Kim, T.K.: Latent-class hough forests for 3D object detection and pose estimation. In: European Conference on Computer Vision. (2014)
10. Doumanoglou, A., Kouskouridas, R., Malassiotis, S., Kim, T.K.: Recovering 6D object pose and predicting next-best-view in the crowd. In: IEEE Conference on Computer Vision and Pattern Recognition. (2016)
11. Akkaladevi, S., Ankerl, M., Heindl, C., Pichler, A.: Tracking multiple rigid symmetric and non-symmetric objects in real-time using depth data. In: IEEE International Conference on Robotics and Automation. (2016)
12. Aldoma, A., Tombari, F., Prankl, J., Richtsfeld, A., Di Stefano, L., Vincze, M.: Multimodal cue integration through hypotheses verification for rgb-d object recognition and 6dof pose estimation. In: IEEE International Conference on Robotics and Automation, IEEE (2013) 2104–2111
13. Kwon, J., Choi, M., Park, F.C., Chun, C.: Particle filtering on the euclidean group: framework and applications. *Robotica* **25**(6) (2007)
14. Chitchian, M., van Amesfoort, A.S., Simonetto, A., Keviczky, T., Sips, H.J.: Adapting particle filter algorithms to many-core architectures. In: Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on, IEEE (2013) 427–438
15. Tan, D.J., Ilic, S.: Multi-forest tracker: A chameleon in tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014)
16. Breiman, L.: Random forests. *Machine learning* **45**(1) (2001)
17. Tjaden, H., Schwanecke, U., Schömer, E.: Real-time monocular pose estimation of 3D objects using temporally consistent local color histograms. In: IEEE Conference on Computer Vision and Pattern Recognition. (2017)

18. Merriaux, P., Dupuis, Y., Boutteau, R., Vasseur, P., Savatier, X.: A study of vicon system positioning performance. *Sensors* **17**(7) (2017) 1591
19. Zhang, Z.: A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(11) (2000) 1330–1334
20. Niehorster, D.C., Li, L., Lappe, M.: The accuracy and precision of position and orientation tracking in the HTC vive virtual reality system for scientific research. *i-Perception* **8**(3) (2017)
21. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv:1602.07360* (2016)
22. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: *IEEE Conference on Computer Vision and Pattern Recognition*. (2017)
23. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015)
24. Kendall, A., Cipolla, R.: Geometric loss functions for camera pose regression with deep learning. In: *IEEE Conference on Computer Vision and Pattern Recognition*. (2017)