A Study on the Impact of Domain Randomization for Monocular Deep 6DoF Pose Estimation

Kelvin B. da Cunha[*], Caio Brito[*†], Lucas Valença[*], Francisco Simões[‡*], Veronica Teichrieb[*]
[*]Voxar Labs, Centro de Informática, Universidade Federal de Pernambuco, Campus Recife, Pernambuco, Brazil
[†]Université de Montréal, Montréal, Québec, Canada
[‡] Curso Técnico em Informática para Internet, Instituto Federal de Pernambuco, Campus Belo Jardim, Pernambuco, Brazil
{kbc, cjsb, lvrma, fpms, vt}@cin.ufpe.br

***Abstract*—In this work, we apply domain randomization to synthetic images and train deep 6DoF monocular RGB pose estimation models to work on a real object. We compare 19 models trained with different combinations of synthetic and real data (fully synthetic, fully real, initially synthetic and supplemented with real, and a real-synthetic randomized mix). By gradually decreasing the amount of real data used, we show it is possible for deep 6DoF detection to obtain superior results while using less real data (which is harder to obtain) and suggest the best approach to train a model with synthetic data. Our method is validated using a textureless 3D printed object, as the textureless category is a challenging, common open problem in itself. A real and a synthetic dataset generated for this work, totalling over 24,800 annotated frames, are also made public. We also show that synthetic, randomized data can help generalize a model by training it to handle challenges such as illumination changes and fast motion. Finally, we also evaluate how a model trained for one camera sensor works with a different one, and show that synthetic simulations of real cameras can help overcoming this challenge.**

I. Introduction

The six degrees-of-freedom (6DoF) pose estimation of rigid objects in an RGB image is a long-time computer vision challenge, with applications in areas such as Mixed Reality (XR) [1], robotics [2], autonomous vehicles [3], human-computer interaction [4], etc.

Recently, detection approaches are quickly improving due to the arrival of deep learning [5], [6]. Models can now be trained to extract relevant information about the object to be detected using only the pixels from input images. These complex RGB deep learning models usually have to be trained with a high amount of labeled images to supplement the lack of depth information available to their RGB-D counterparts [7], [8].

The main issue arises from the fact that filming those training sequences as well as annotating their ground truth is generally a cumbersome, expensive, and time-consuming task. Challenges include setting up and calibrating markers and cameras to properly capture the ground truth [9]–[11], as well as dealing with problems such as camera noise, occlusions, and motion blur. Moreover, a wide array of viewpoints and rotation angles must be covered, all while accounting for background, foreground, and illumination details. Even then, detection trained using those sequences might not work well with different cameras or environments.

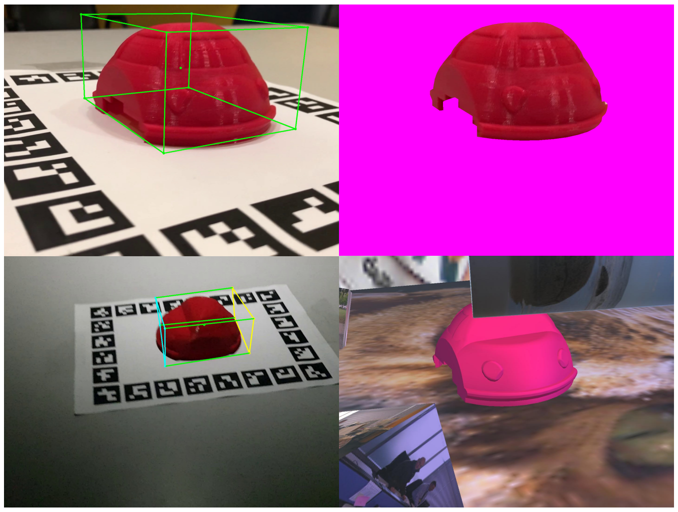


Fig. 1. Clockwise from the top-left corner: real data acquisition with ground truth pose (in green); segmented object from real frame; data generated through domain randomization; detection output in a challenging motion and illumination scenario for a model trained with both real and synthetic data.

For these aforementioned reasons, it is highly desirable to be able to generate such training sequences artificially, using just a computer aided design (CAD) model, and have that work for the detection of real objects. Yet, that remains an open challenge due to the reality gap. Our work proposes a way to introduce configurations of diverse synthetic data to decrease the number of real images required during training, while simultaneously improving the model's response when faced with detection challenges.

In the subject of application fields for RGB detection, one relevant area for both the industry and the academy is the development of AR and VR applications using 3D printed objects [12], [13], which are becoming more common every day. As these objects are available both in their digital (pre-print) and physical (printed) forms, they are widely suitable for an multiple user-end XR applications [14]. Using these objects' CAD models, synthetic data can be generated to train a network even before they are printed. This can make the process of detecting a new object much quicker and user-friendly. It is worth noting that although these objects' lack of texturization makes them very suitable for AR, it also makes them more challenging, as they lack distinctive features and can vary widely in terms of color, size, and specularity.

This work's main contribution is evaluating how the introduction of synthetic data with domain randomization (DR) can improve 6DoF detection, both in terms of precision and ease of training. Specifically, we investigate the impact of such points on the 6DoF detection of a 3D printed, textureless object. The DR approach is evaluated to measure improvements in model accuracy when faced with generic real-world challenges (e.g., different illumination, motion, and different camera sensors). This is performed while evaluating how results are altered by the balance between synthetic and real training data distributions. This way, we can obtain clues as to which scenarios and challenges benefit more from either synthetic or real training. An application to easily render and annotate synthetic images for a general-purpose detection scenario is also proposed. To the best of our knowledge, this is the first work to investigate the impact of DR to increase model robustness to camera hardware changes (i.e., when the model is trained with one camera but tested with a different one). Having robustness to different cameras is especially relevant to the industry, as it must be handled when attempting to develop generic models and applications for a wide array of user-end devices. For an overview of our data generation results and detection output, see Fig. 1.

II. Related Work

A. 6DoF Object Pose Estimation

Recently, many robust markerless 6DoF pose estimation approaches have been proposed. Out of these, RGB-D techniques have received special attention [15], [16]. These works obtain a very high accuracy due to the RGB input frame being supplemented by depth maps. Yet, those depth maps are limited by the view distance, and struggle with illumination such as sunlight. In those cases, RGB-D techniques are prone to fail, as they are left with just the RGB data. Considering how much less common and more expensive RGB-D sensors are compared to standard RGB camera, multiple works have also attempted to solve this issue using purely RGB [1], [8].

Traditionally, techniques have used image processing and geometric concepts to tackle this problem. Region-based works [1], [17] evaluate the color distribution in image regions. Edge-based works attempt to match the CAD model's edges to the RGB input's gradients [18]. Feature-based use keypoints and descriptors [19], [20]. Other techniques attempt to use traditional machine learning algorithms to interpret extracted features and optimize a hypothesis [21], [22]. Finally, there are hybrid approaches that merge the aforementioned geometric concepts with machine learning [9], [15], [23].

In recent years, deep learning approaches have been introduced. Those can learn about the object of interest by being trained with example images, without needing predefined features or clues like traditional approaches do. This characteristic makes deep learning especially suitable to perform pose estimation in scenarios where there's not much available information, such as plain RGB. Though it has been shown to work for real-time object tracking (with temporal consistency) [24], [25], most deep learning works attempt to perform object detection on a frame by frame basis [7], [8], [23], [26]. Newer deep learning approaches have been using more complex, deeper pipelines to perform both pose prediction and refinement directly [5], [25].

On the other hand, some approaches simplify the problem by attempting to detect image keypoints [20] or bounding box corners [8], [26] and only then perform minimization calculations in a more traditional way, such as by solving least-squares problems or similar algorithms. The simplified architectures of such approaches make training the model much easier, while still being able to achieve high accuracy values. However, they still suffer from a major drawback: the need for large amounts of annotated data. Deep learning 6DoF works have been shown to be able to detect synthetic objects when trained with synthetic data, but most still require real data to detect real objects.

B. Synthetic Dataset Generation

As explained in Section I, capturing and annotating real datasets are an expensive, time-consuming task with lots of variables and prone to constant error at multiple stages. This is especially true for 6DoF, as the ground truth is hard to capture, given the higher complexity of the information. Yet, deep learning approaches still require large amounts of real data to work properly. This is due to the reality gap between virtual and real images, which makes it hard for the network to learn about the real object just by seeing a virtual version of it.

To tackle this reality gap, Tobin et al. [27] explored the concept of domain randomization of synthetic frames. Using it, they trained a neural network with the task of localizing simple, colorful geometric shapes on top of a table for robotic manipulation. This work also provided a robotics-oriented localization training dataset with high variability to the neural network that can generalize to real-world data. The dataset generator was built using the MuJoCo Physics Engine [28] and generates images by randomizing the positions and textures of the object of interest and surrounding objects, as well as randomizing illumination, camera position, distractor configuration, and the type and amount of noise present on the images. This work, albeit performing a task simpler than 6DoF pose estimation, is very relevant as the system can achieve high accuracy (1.5 cm) without any pre-training on real images.

Tremblay et al. [3], from NVIDIA, also used DR to present a system for training deep neural networks for 2D car detection on outdoor scenarios. The scenes were generated using Unreal Engine (UE4) by changing randomly the number, type, and texture of the objects, camera translation and rotation, illumination, background, and floor textures, and the set of distractors, which were either *contextual distractors* (objects similar to possible real scene elements, positioned randomly but coherently) or *flying distractors* (geometric shapes with random texture, size, and position). Their network presented better results when using the synthetic DR dataset mixed with

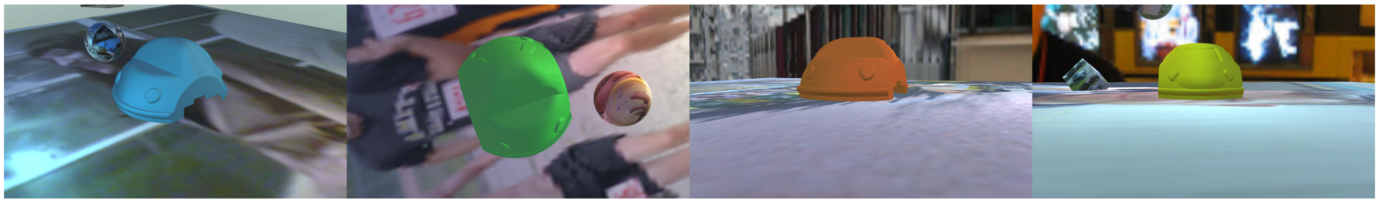


Fig. 2. Example of synthetic frames generated using the domain randomization process from Section III-A.

real images. Other alternatives tested were using only real images and using a mix of real and realistic (non-randomized) synthetic data from the VKITTI dataset [29]. Even though this work also does not perform 3D detection or 6DoF estimation, it is a step forward in this direction, as the complexity of the objects and scenes is very significant.

NVIDIA also published their Deep learning Dataset Synthesizer (NDDS) [30], a UE4 plugin that can generate synthetic data in real-time by randomizing illumination, objects, cameras, poses, textures, and distractors. The plugin can export images, segmentations, depth maps, object pose annotations, bounding boxes, keypoints, and custom stencils. The system was used to create several datasets. The Falling Things dataset (FAT) [31] uses NDDS and focuses on the context of 6DoF object detection. Each of its photorealistic images consists of a stereo frame pair with corresponding depth images, 3D poses, bounding boxes, and the segmentation image of each scene's objects. The objects were extracted from the YCB dataset [32] and used in 3 virtual environments within UE4 in random positions that change after the objects fall into the scenes. Yet, it does not perform DR.

The Synthetic Image Dataset for 3D Object Pose Recognition with Distractors (SIDOD) dataset [33] is similar to the FAT dataset but with the presence of flying distractors (see [3]) and domain randomization. Lee at al. [34] extended NDDS to handle robotic joints and export the joint information to be used in network training. Still, all these NDDS datasets are synthetic, without similar real frames to compare against, so they are best suited for fully synthetic approaches.

Hinterstoisser et al. [35] proposed a strategy that generates cluttered synthetic frames and trains a network to perform 2D detection of complex objects. Though the scenes are cluttered, the technique guarantees that the objects of interest are presented equally to the neural network in all scenarios. The model trained with only synthetic data outperformed the model trained with only real data. The authors also investigated individual aspects of the image generation pipeline, which revealed that blur and illumination color are the scene aspects that influence results the most.

In the 6DoF context, Tremblay et al. [2] proposed a DOPE deep learning approach that infers the 6DoF poses of known objects from a single RGB image without requiring refinement. The system was trained using a combination of photorealistic images from the FAT dataset [31] and non-photorealistic domain randomized frames with varying types of flying distractors (cones, pyramids, spheres, cylinders, partial toroids, arrows), randomized illumination, occlusion, and distractor and background textures. Their DOPE approach had comparable performance to a state-of-the-art network trained using a mix of real and synthetic data and was the first deep 6DoF work to achieve such precision levels with only synthetic data. In our work, we aim to deepen the understanding of how these data distributions influence a network. In particular, how DR alone can make networks easier to train and more robust to challenges. The motivation being that photorealistic frames (such as the ones from the FAT dataset) are hard to generate and can be comparable in difficulty and time consumption to recording and annotating real data, depending on the challenge to be simulated. Thus, it is important to understand how the simpler way (domain randomization) alone can impact results.

III. Method Overview

This section describes the processes of synthetic data generation, real data acquisition, and 6DoF detection. All the over $20,000$ real and $4,000$ synthetic annotated frames generated in this work have been made available as public domain[1].

A. Synthetic Data Generation

Our synthetic data generation process varies color, textures, background, occlusion, illumination, and viewpoint. The intention was to expose the object to a wide array of variations, each having their own contribution to the final result, as explained below.

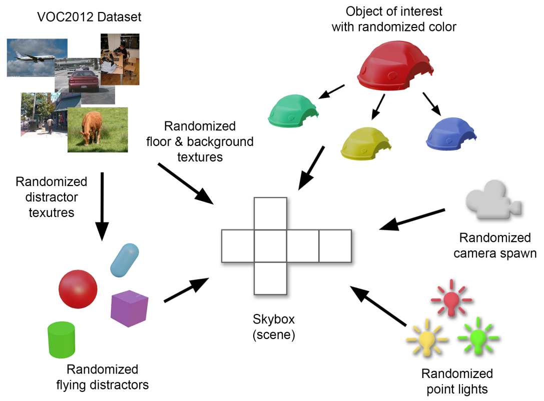

VOC2012 Dataset
Object of interest with randomized color
Randomized floor & background textures
Randomized distractor texutres
Randomized camera spawn
Skybox (scene)
Randomized flying distractors
Randomized point lights

Fig. 3. Diagram illustrating the domain randomization and synthetic scene generation process as described in Section III-A.

The synthetic scenes used in this work all had one basic structure for randomization: 1 object over a flat surface (floor),

[1]https://github.com/Kelvin-Cunha/3dprinted-objects-6dof

1 camera, up to 5 *flying distractors* and up to 15 point lights. All objects were contained within a *skybox*. The cameras used in our work had their output size and intrinsic parameters (focal length and principal point in pixels) configured to simulate the ones obtained by calibrating the cameras used during real data acquisition (see Section III-B). Radial and tangential lens distortion were not simulated. As documented in Kehl et al. [7], randomizing the camera's position increases robustness to multiple viewpoints. Our camera was always pointing at the origin (where the object of interest is fixed), and its randomized spawn location (always contained within the skybox cube, which is $50m$ in size) has radial distance limited from $4m$ to $10m$, azimuth angle from $0°$ to $180°$, and polar angle from $0°$ to $360°$, following the spherical coordinate system. Those ranges were set keeping in mind our indoor, small object scenario.

Tremblay et al.'s work [3] illustrates the importance of randomizing the textures of the background (skybox) and floor (surface beneath the object) to avoid learning features that do not belong to the object itself. It also suggests adding *flying distractors*, simple 3D geometric objects of varying textures, in random places. Our flying distractors consisted of cubes, spheres, cylinders, and capsules of a fixed size. Those distractors further increase robustness to occlusions, as they can sometimes partially occlude the object. In fact, unlike [3] (that deals with vehicles), our small object scenario had cases where total or near-total occlusions happened, making it necessary for us to introduce a new step to the randomization: if the object has more than 35% of its 2D area occluded in the final render (value based on [11] experiments), the frame is discarded. All skybox, floor, and distractor textures were taken from the Visual Object Classes (VOC) dataset [36], which contains photos of scenes with elements such as clutter and other objects (as opposed to the outdoors backgrounds used by Tremblay et al.). This choice provided a more realistic scenario for our indoors use case. As [35] has stated the importance of blur and illumination, we have trained models synthetically for illumination but not for blur, and added a test scenario for each of the 2 to evaluate generalization. To increase robustness to illumination, Tremblay et al. suggests randomizing the lights of the scene. Our indoor scenario used point lights of random colors with constant radius and intensity, spawned at random locations inside the skybox. Finally, [3] suggests that randomizing the object of interest's texture is desirable, as it makes the network learn the object's structure as opposed to its appearance. Given our textureless scenario, the object's color was randomized, but no texture images were added. The entire process is illustrated in Fig. 3. Our synthetic dataset generation tool was developed in C# using the 3D engine Unity.

B. Real Data Acquisition

For every frame of our real RGB sequences, the object was placed on top of a field of ArUco markers [37] in order for the 6DoF ground truth to be obtained. By projecting the object's CAD model using the 6DoF pose obtained through the marker field, we could segment only the pixels which corresponded to the object, removing the background and building a binary segmentation mask. The segmented frame was then matched to a random background from the VOC dataset [36]. This was necessary to avoid overfitting, given the constant marker field below the object. The projected object 3D model's depth map is also stored. This is a way RGB techniques are trained to supplement the lack of RGB-D data. Finally, using the 6DoF pose, the 8 corners of the object's 3D bounding box can be projected on the frame and stored together with the segmentation mask, depth map, and new-background (see Fig. 4). Our real data acquisition tool was developed in C++ and uses OpenCV and OpenGL to perform the described tasks.

The sequences recorded contain a wide array of viewpoints and camera rotation, motion blur, and illumination challenges such as a moving point light in a dark room, indirect sunlight, and varying indoor illumination patterns.

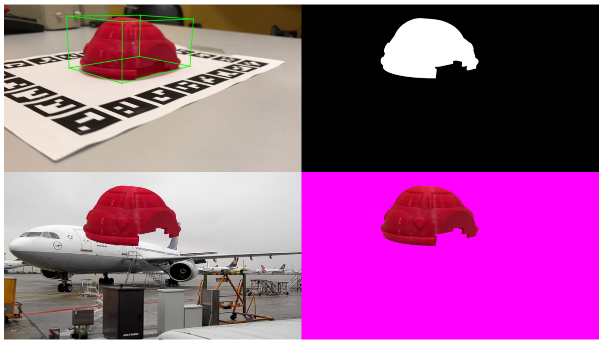


Fig. 4. Real data generation process. Clockwise from the top-left corner: raw RGB frame and object bounding box (in green) projected according to the ground truth pose from the ArUco marker field, binary mask, segmented object, new-background random frame the VOC dataset.

C. 6DoF Detection

In order to evaluate the impact different training data, a model that performed monocular RGB 6DoF detection had to be chosen. We opted to use the supervised estimator proposed by Tekin et al. [8] because it is robust, simple to train, and publicly available[2].

IV. Real Dataset Generation

To map real-world challenges and test the proposed method, sequences were recorded and processed according to Section III-B. These sequences were captured with two different mobile cameras, to enable the assessment of how a model trained with one camera performs with another, which to our knowledge has not been previously investigated in other works. It is also worth to mention again our 3D printed textureless focus, which is a difficult area for the object's lack of features. The 3D object chosen was printed using a solid-color red Polylactic Acid (PLA) filament, which is a very common configuration for 3D prints, and which interacts considerably with the scene's illumination. The generated sequences, camera calibrations, and CAD model have also

[2]https://github.com/Microsoft/singleshotpose

been made available as public domain (see Footnote 1). We believe this collateral contribution can help to close those gaps in the literature.

To capture the frames at 1920×1080 resolution, we used the back-facing cameras of an Apple iPhone X and a Samsung Galaxy S8. Sequences listed below are organized by the camera model and present challenges.

- **X_Simple**: Scene recorded using the iPhone X. The camera circles around the object that is fixed at the center of a marker field as per Fig. 4. Movements are slow and controlled, indoor illumination is fixed;
- **X_Motion**: Same as X_Simple but with faster camera motion, introducing motion blur and shaking;
- **S_Simple**: Same as X_Simple, but recorded with the Galaxy S8;
- **S_Motion**: Same as X_Motion, but recorded with the Galaxy S8;
- **S_Illumination**: Same as S_Simple, but with changing illumination slightly;
- **S_Extreme**: Challenging illumination scenario recorded with the Galaxy S8 and with faster camera motion. The same scene as S_Motion but with all indoor lights turned off, the room completely dark, and the object being illuminated by a small, moving point light.

V. Evaluation

Training and testing were ran on a desktop computer with a quad-core CPU @ 3.60 GHz, 32 GB of RAM, and an NVIDIA GeForce GTX 1080 Ti GPU.

A. Datasets

Sequences X_Simple and S_Simple from Section IV were divided in separate portions for training (70%, $\simeq 3,500$ frames) and testing (30%, $\simeq 1,500$ frames). The remaining, more complex scenes were used exclusively for testing to enable us to pinpoint exactly which challenges (unseen by the model during training) were (or not) troublesome. It also enables us to train the model to learn how to deal with these challenges solely by using synthetic, randomized data to evaluate its influence.

B. Model Training

The model was always trained for 500 epochs (fixed). An Stochastic Gradient Descent (SGD) optimizer was used with learning rate $\alpha = 0.001$, decreasing tenfold every 100 epochs. The original model input process resizes frames to 680×680 pixels, which was used for testing. Yet, during training as per [8], in order to increase robustness to scale changes, frames were additionally scaled uniformly in width and height by a random factor of 32 between 320 and 680. Input camera parameters are also changed accordingly.

A total of 19 models were trained and evaluated in this work. They are listed below, organized by the data used to train them. A total of $4,000$ synthetic frames (with domain randomization) was used in addition to the $\simeq 7,000$ total of available real frames from X_Simple and S_Simple.

- **Full_X:** Trained only using real images from X_Simple;
- **Full_S:** Trained only using real images from S_Simple;
- **Full_DR:** Only with domain-randomized frames;
- **Mix_X10, Mix_X30, Mix_X60, Mix_X100:** Trained mixing the batches with real and synthetic frames at random. Used 10%, 30%, 60%, and 100% of X_Simple, respectively, in addition to 100% of the synthetic set.
- **Mix_S10, Mix_S30, Mix_S60, Mix_S100:** Same as above, but images came from S_Simple.
- **Fine_X10, Fine_X30, Fine_X60, Fine_X100:** Trained initially using exclusively 100% of the synthetic set of frames, then afterwards fine-tuned (supplemented) with 10%, 30%, 60%, and 100% of the frames from X_Simple, respectively.
- **Fine_S10, Fine_S30, Fine_S60, Fine_S100:** Same as above, but images came from S_Simple.

C. Evaluation Metrics

The pose error predicted by the models was measured using 3 different metrics, described below.

- **Reprojection Error (Rep.)**: Prediction is accepted if the mean 2D Euclidean distance (in pixels) for all mesh vertices projected using the predicted pose and the ground truth is less than 5, as per [8].
- **3D Pose Error (ADD):** Described as the average 3D Euclidean distance between all mesh vertices multiplied by the 6DoF pose transformations corresponding to the ground truth and model prediction. As per [10], a prediction is accepted if the ADD value is less than 10% of the model's 3D diameter, calculated as the maximum distance between two mesh vertices.
- **Pose Accuracy (Pose):** In this case, translation and rotation errors are measured individually. A prediction is accepted if its translation and rotation difference to the ground truth is less than 5 cm and is less than $5°$ respectively [38].

D. Experimental Results

Results using the models described in Section V-B with the datasets from Section V-A under the metrics from Section V-C can be seen in Table I. Further evaluation of representative models by variating the ADD threshold can be seen in Fig. 5.

VI. Discussion

A. Full Models

As expected, Table I shows that the models trained fully with real data perform best in scenarios similar to the ones they were trained with (S_Simple and X_Simple). There is about 90% or more of a drop in precision when new challenges are introduced (S_Motion, S_Illumination, S_Extreme, X_Motion). It is also possible to notice that camera sensor variation has an equally high impact on the final performance of the model, even when under the same scenario (Full_X in S_Simple and Full_S in X_Simple). This is a significant problem for generic real-world applications where end-user configurations are not known and cannot be fully mapped.

TABLE I
PERCENTAGE OF ACCEPTED FRAMES PER METRIC FROM SECTION V-C FOR ALL MODELS USING ALL SEQUENCES, AS DESCRIBED IN SECTIONS V-B AND V-A. HIGHER IS BETTER. UNDERLINED VALUES REPRESENT LOCAL BEST RESULTS, BOLD VALUES REPRESENT OVERALL BEST RESULTS.

Model	S_Simple			S_Motion			S_Illumination			S_Extreme			X_Simple			X_Motion		
	Rep.	ADD	Pose	Rep.	ADD	Pose	Rep.	ADD	Pose	Rep.	ADD	Pose	Rep.	ADD	Pose	Rep.	ADD	Pose
Full_X	4.55	7.14	6.48	0.00	0.88	0.00	0.00	0.70	0.00	0.00	0.00	0.00	26.22	18.80	54.90	3.26	1.38	0.10
Full_S	41.61	25.67	46.27	1.91	3.80	0.00	0.00	1.53	0.00	0.00	0.00	0.00	0.00	3.95	1.74	0.00	0.12	0.00
Full_DR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mix_X10	7.45	9.07	7.14	0.00	0.00	0.00	0.00	2.29	0.00	0.00	0.00	0.00	26.86	33.10	56.37	7.07	7.07	1.12
Mix_X30	7.86	6.83	9.63	0.00	1.40	0.71	0.00	1.76	1.76	0.00	0.00	0.00	21.48	34.57	64.93	7.27	5.93	2.71
Mix_X60	7.45	14.60	9.94	0.00	0.78	0.00	0.00	3.44	1.67	0.00	0.00	0.00	26.46	**48.66**	**83.10**	7.07	6.09	2.03
Mix_X100	7.10	14.05	9.35	0.00	1.50	2.00	0.00	2.95	1.36	0.00	0.13	0.00	28.41	29.61	22.26	7.07	5.22	4.98
Mix_S10	8.70	11.18	11.18	0.00	0.75	0.07	0.00	0.57	0.00	0.00	0.13	0.00	12.29	14.58	12.37	0.00	0.12	0.11
Mix_S30	36.65	39.75	58.07	0.00	1.67	0.71	0.48	4.01	4.39	0.00	0.26	0.00	10.51	15.32	10.27	0.00	1.60	3.45
Mix_S60	33.85	28.57	51.86	0.00	2.50	1.70	2.53	4.59	4.77	0.00	0.13	0.00	16.08	15.93	15.79	5.11	7.87	4.58
Mix_S100	20.50	17.39	46.27	0.00	4.10	4.10	0.57	2.67	6.30	0.00	0.13	0.00	10.90	15.64	13.98	2.74	7.07	6.47
Fine_X10	20.19	14.91	25.78	3.39	6.58	4.81	2.70	8.02	3.24	6.28	5.13	7.95	34.36	12.01	38.70	10.32	10.74	10.55
Fine_X30	27.33	33.85	44.72	6.37	11.93	9.02	2.70	**13.93**	4.20	8.46	8.46	9.23	38.70	28.36	76.38	10.28	17.23	17.60
Fine_X60	28.88	27.33	34.16	10.17	**15.25**	**9.69**	2.70	10.50	2.67	8.33	7.31	8.08	38.55	27.09	75.91	11.87	17.75	16.00
Fine_X100	17.70	24.53	36.65	5.22	11.53	4.34	2.70	11.26	4.58	6.28	8.21	3.95	**43.68**	29.70	79.70	**11.90**	**18.63**	**18.26**
Fine_S10	52.80	35.09	65.22	6.03	6.85	5.36	2.67	8.02	**11.26**	**9.36**	12.44	12.69	13.35	19.67	19.27	5.11	7.69	9.19
Fine_S30	55.59	**48.14**	79.81	6.58	7.32	5.36	1.53	**13.93**	8.21	4.74	24.74	13.08	20.93	21.01	19.19	4.77	7.40	6.47
Fine_S60	**63.98**	36.65	**87.58**	**13.08**	7.39	9.02	**3.05**	10.50	9.35	8.97	**24.87**	12.31	15.09	18.80	22.91	9.41	9.73	11.18
Fine_S100	55.59	45.84	76.71	10.98	7.93	8.61	1.15	8.21	3.63	6.79	20.38	**13.46**	11.22	14.22	19.19	9.24	7.87	10.37

Likewise, the model trained fully with randomized data (Full_DR) does not have a satisfactory performance at the tests with real data, even in the most straightforward scenarios. This behavior is expected due to the large difference between the non-photorealistic randomized synthetic scenarios and the real world, causing a huge reality gap. In fact, some existing attributes of the original object could not be mapped in the frames rendered, such as object texture, color, and deformations caused in the printing process. This makes the difference between the distributions generated through real and synthetic data be even greater, in addition to the influence of the reality gap and variations that existing on sensors used in real cameras. These mentioned factors make our synthetic data generation process of little use when used alone to evaluate real scenarios. However the DR information can be used to guide the training to achieve a better result making the synthetic data to perform models adjustments and regularization on training. For this reason, we carry out the next experiments with combinations of real and synthetic data hoping that the use of synthetic data will help to avoid overfitting for the real images on the specific training scenario and improve model generalization on different data distributions.

It is also possible to note that all models failed in the S_Extreme scenario. Even the model trained using the same camera (Full_S) showing how serious the generalization problems are when there is a change in motion and illumination conditions. Those scenarios are very common especially in mobile, as the devices often go to different locations with different illuminations and are constantly in motion.

B. Mixed Models

As we can see in Table I, mixing synthetic and real data produces a slight improvement for test cases with the same properties as the images used in training (Mix_S30 and Mix_S60 in S_Simple, all Mix_X in X_Simple). This strategy reduces the reality gap issues for the model trained solely with synthetic data (Full_DR) while improving generalization and reducing overfitting with respect to the ones trained with only real data (Full_X, Full_S).

This generalization effect also improves performance with different camera hardware under similar scenarios to the ones seen training (all Mix_S with X_Simple, all Mix_X with S_Simple). Results are still about 25 to 50% lower than with the original camera, but it is a considerable improvement from the results in Section VI-A. Results in more complex scenarios with variations in lighting and movement were slightly better, but not significantly. This is expected as S_Motion, X_Motion, and S_Extreme introduce motion blur, which was not added to the randomized training nor the real training sequences. It is also interesting to point out that Mix_S60 has obtained the best results from both the mixed and full models when handling minor illumination changes, a challenge that was present in the randomized data. It is also important to consider that training was more challenging here, given the simplicity and lack of features of the object of interest, coupled with the relatively low number of frames used for each model and the high amount of randomization in the synthetic scenes. Training in this mixed manner with a much higher number of frames, as performed by [2], might further improve results.

Overall, results show that mixing synthetic and real data obtains results that are either comparable or superior, showing that it is possible to train networks more easily with smaller amounts and simpler scenarios of real data, mixed with random synthetic frames. Even if improvements are not yet significant in precision, being easier and faster while remaining at least comparable is an interesting outcome.

C. Fine-Tuned Models

This scenario covers models trained initially with synthetic data and then supplemented (fine-tuned) with real data. As mentioned in section V-B the synthetic pretraining is performed using all synthetic set available and later models are adjusted using different percentages of the training set with real images. It was (by far) the best alternative, obtaining the best overall results in 16 out of the 18 scores and being comparable in the remaining 2. It is interesting to note that the best variations were using 30 and 60% of real data as supplementation, showing that not necessarily more real information is beneficial.

Moreover, the performance numbers obtained with different camera hardware (e.g., all Fine_X when compared to Full_X and all Mix_X on the scene S_Simple) shows that this form of training is an effective way to simulate variations in camera sensor and gain robustness to this challenge, which was mostly overwhelming to the other approaches from Sections VI-A and VI-B. The fine-tuned approach also had the best results in the scenes with fast motion (X_Motion, S_Motion, and S_Extreme), showing that randomization is able to somewhat generalize the model even when the motion blur challenge itself is not present in the synthetic data. The approach also outperformed the others for illumination challenges.

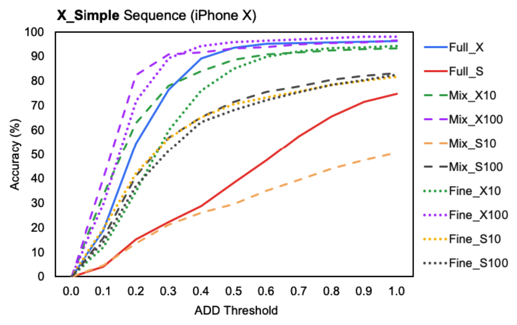
Results indicate how crucial model initialization is to the training process. They also display how well synthetic, randomized data can accelerate convergence, requiring less real data and being more efficient in avoiding overfitting than the approaches mentioned in Sections VI-A and VI-B.

D. *ADD Threshold Variation*

Figure 5 gives a general idea of how models behave (in the simple scenarios X_Simple and S_Simple) as the ADD threshold increases, accepting poses that are farther away from the ground truth. This can give an indication of which model is, overall, predicting values closer to the ground truth. Though such values may be still unacceptable in terms of real-world usage for 6DoF detection, they can give indications of which variations show more promise to be trained further.

Note that models fine-tuned with just 10% of real data perform better overall than most of those trained with another camera. It is thus visible in the charts that models trained with the same camera as the sequence have a general advantage. That is especially true for the iPhone X chart, where same-camera models are also able to reach higher accuracies faster ($10 \sim 20\%$ against $20 \sim 40\%$ of the S8). This might be an indication of how much a higher-end camera influences training (even with the same image resolution as the S8). This hardware caveat is especially valid because while a value around 10% might be acceptable, for most applications, $20 \sim 40\%$ certainly is not. Yet, the fine-tuning approach's efficacy is also validated, as the iPhone X's fine-tuned results have outperformed most mixed results for the Galaxy S8 in its own chart. Finally, we want to once more draw attention to how synthetic data, and how the data was introduced, have alleviated sensor differences (which were shown to be a significant challenge). This is an indicator of how to train more generic user-focused models in the future.

Because we have shown that initializing the model with synthetic data is highly preferable, we would like to further suggest that, for our 3D printed scenario, a model could begin to be trained synthetically while the 3D object is still being printed. The pre-trained estimator can then be fine-tuned afterward by a small set of images (which can be captured by the user's camera sensor), obtaining superior precision. This can be a much quicker and cost-effective way to detect 3D objects, as opposed to trying to generate a very large real or photorealistic dataset. Is would also be better suited to the user's camera lens, and possibly robust to different, challenging scenarios due to randomization.



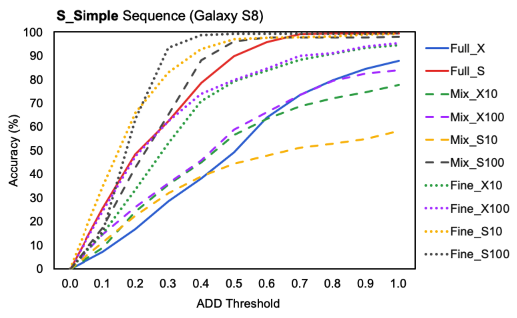


Fig. 5. Results for representative models when varying the ADD threshold. Top chart shows results for the X_Simple sequence, bottom chart S_Simple.

VII. Conclusion

In this work, we have conducted an extensive analysis to evaluate how different training strategies a model with DR impacted the output of a 6DoF pose estimator. We have included numeric analyses and discussions on little-explored, relevant aspects such as variations in camera sensor and illumination. Results have shown that initializing a model with synthetic, randomize data can improve its convergence even with a low number of input frames. They have also shown that more real data is not necessarily a good thing. Our results indicate that synthetic data can better prepare a model for different real scenarios, especially in an industry case of many different users with different cameras. Finally, we have made available two datasets, one real and one synthetic, which use the same textureless object and can be combined for training. A more efficient way to train models to detect 3D printable objects was also suggested as a consequence of the obtained results. For future work, we must perform a careful ablation study to identify better how each DR aspect impacts 6DoF. It would also be interesting to reproduce this work, but using photorealistic synthetic data instead of real data. More complex challenges such as occlusions, same-color clutter, outdoor illumination, noise, and varying environments also need evaluation. For those, we must further improve our data generation tools. Testing with a wider array of objects (with geometries of increasing complexity) and camera sensors (of different qualities) is also desirable.

References

[1] H. Tjaden, U. Schwanecke, E. Schömer, and D. Cremers, "A region-based gauss-newton approach to real-time monocular multiple object tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1797–1812, 2018.

[2] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," *arXiv preprint arXiv:1809.10790*, 2018.

[3] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 969–977.

[4] P.-C. Wu, R. Wang, K. Kin, C. Twigg, S. Han, M.-H. Yang, , and S.-Y. Chien, "Dodecapen: Accurate 6dof tracking of a passive stylus," 2017.

[5] S. Zakharov, I. Shugurov, and S. Ilic, "Dpod: 6d pose object detector and refiner," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1941–1950.

[6] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570.

[7] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," in *Proceedings of the International Conference on Computer Vision (ICCV 2017), Venice, Italy*, 2017, pp. 22–29.

[8] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.

[9] A. Tejani, R. Kouskouridas, A. Doumanoglou, D. Tang, and T.-K. Kim, "Latent-Class Hough Forests for 6-DoF Object Pose Estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 1, pp. 119–132, 2018.

[10] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Asian conference on computer vision*. Springer, 2012, pp. 548–562.

[11] M. Garon, D. Laurendeau, and J.-F. Lalonde, "A framework for evaluating 6-dof object trackers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 582–597.

[12] T. H. Jung and M. C. tom Dieck, "Augmented reality, virtual reality and 3d printing for the co-creation of value for the visitor experience at cultural heritage places," *Journal of Place Management and Development*, 2017.

[13] V. Carlota. Gartner hype cycle 2019: 3d printing predictions. [Online]. Available: https://www.3dnatives.com/en/gartner-hype-cycle-3dprintingpredictions-150120194/#!

[14] L. Wang, "Method for instructing a 3d printing system comprising a 3d printer and 3d printing system," U.S. Patent US9 776 364B2, 10 03, 2017.

[15] D. J. Tan, N. Navab, and F. Tombari, "Looking beyond the simple scenarios: combining learners and optimizers in 3D temporal tracking," *IEEE Transactions on Visualization & Computer Graphics*, no. 1, pp. 1–1, 2017.

[16] M. Tian, L. Pan, M. H. Ang Jr, and G. H. Lee, "Robust 6d object pose estimation by learning rgb-d features," *arXiv preprint arXiv:2003.00188*, 2020.

[17] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6d object pose and size estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2642–2651.

[18] S. Trinh, F. Spindler, E. Marchand, and F. Chaumette, "A modular framework for model-based visual tracking using edge, texture and depth features," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 89–96.

[19] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6d object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[20] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[21] C. Sahin and T.-K. Kim, "Category-level 6D Object Pose Recovery in Depth Images," *arXiv preprint arXiv:1808.00255*, 2018.

[22] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6D object pose and predicting next-best-view in the crowd," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3583–3592.

[23] K. Park, T. Patten, and M. Vincze, "Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[24] M. Garon and J.-F. Lalonde, "Deep 6-DOF tracking," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 11, pp. 2410–2418, 2017.

[25] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 683–698.

[26] C. Song, J. Song, and Q. Huang, "Hybridpose: 6d object pose estimation under hybrid representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 431–440.

[27] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[28] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[29] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[30] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, W. Hodge, and S. Birchfield, "NDDS: NVIDIA deep learning dataset synthesizer," 2018, https://github.com/NVIDIA/Dataset_Synthesizer.

[31] J. Tremblay, T. To, and S. Birchfield, "Falling things: A synthetic dataset for 3d object detection and pose estimation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[32] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 International Conference on Advanced Robotics (ICAR)*, 2015, pp. 510–517.

[33] M. Jalal, J. Spjut, B. Boudaoud, and M. Betke, "Sidod: A synthetic image dataset for 3d object pose recognition with distractors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[34] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield, "Camera-to-robot pose estimation from a single image," *arXiv preprint arXiv:1911.09231*, 2019.

[35] S. Hinterstoisser, O. Pauly, H. Heibel, M. Marek, and M. Bokeloh, "An annotation saved is an annotation earned: Using fully synthetic training for object detection," in *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*. IEEE, 2019, pp. 2787–2796. [Online]. Available: https://doi.org/10.1109/ICCVW.2019.00340

[36] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

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

[38] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold *et al.*, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3364–3372.