

## RESEARCH

# Uncovering the Background-Induced bias in RGB based 6-DoF Object Pose Estimation

Elena Govi<sup>1,2</sup>, Davide Sapienza<sup>1,2</sup>, Carmelo Scribano<sup>1,2</sup>, Tobia Poppi<sup>1</sup>, Giorgia Franchini<sup>1\*</sup>, Paola Ardón<sup>3</sup>, Micaela Verucchi<sup>4</sup> and Marko Bertogna<sup>1,4</sup>

\*Correspondence:

[giorgia.franchini@unimore.it](mailto:giorgia.franchini@unimore.it)

<sup>1</sup>Department of Physics, Informatics and Mathematics, University of Modena and Reggio Emilia, Modena, IT

Full list of author information is available at the end of the article

## Abstract

In recent years, there has been a growing trend of using data-driven methods in industrial settings. These kinds of methods often process video images or parts, therefore the integrity of such images is crucial. Sometimes datasets, e.g. consisting of images, can be sophisticated for various reasons. It becomes critical to understand how the manipulation of video and images can impact the effectiveness of a machine learning method. Our case study aims precisely to analyze the Linemod dataset, considered the state of the art in 6D pose estimation context. That dataset presents images accompanied by ArUco markers; it is evident that such markers will not be available in real-world contexts. We analyze how the presence of the markers affects the pose estimation accuracy, and how this bias may be mitigated through data augmentation and other methods. Our work aims to show how the presence of these markers goes to modify, in the testing phase, the effectiveness of the deep learning method used. In particular, we will demonstrate, through the tool of saliency maps, how the focus of the neural network is captured in part by these ArUco markers. Finally, a new dataset, obtained by applying geometric tools to Linemod, will be proposed in order to demonstrate our hypothesis and uncovering the bias. Our results demonstrate the potential for bias in 6DOF pose estimation networks, and suggest methods for reducing this bias when training with markers.

**Keywords:** Digital image; Data manipulation; Linemod; Convolutional Neural Networks; Deep Learning explainability; 6DoF pose estimation; Saliency maps

## 1 Introduction

### 1.1 Domain of interest

Recovering the **6-Degrees-of-Freedom** (often referred to as **6D**) pose of an object from a single RGB image is a relevant computer vision problem with several applications in the domains of industrial automation [1], robotics [2] [3], automotive [4] [5], augmented reality [6] and several others. Generally speaking, given an RGB image  $I \in \mathbb{R}^{w \times h \times 3}$ , a **6D** pose estimation algorithm should recover the translation  $\mathbf{t} = (t_x, t_y, t_z)$  and rotation  $\mathbf{R} = (r_x, r_y, r_z)$  vectors that describe the position and orientation of an object in the camera coordinate system [7]. Recent learning-based techniques that use **Deep Neural Networks** (DNNs) have proven to achieve very high performance scores in the **6D** object pose estimation, achieving state-of-the-art results. **Convolutional Neural Networks** (CNNs) in particular, achieve impressive results in a wide variety of computer vision problems, including **6D** pose estimation from RGB input, at the cost of being extremely data driven and requiring

huge amounts of labeled training examples. In Section 2.1 we present a taxonomy of the most popular Deep Learning methods for 6D object pose estimation. While for other computer vision tasks the dataset acquisition and labeling are easy to obtain, resorting to manual labeling, this is not the case for 6D object pose, since identifying ground truth translations and rotations from real images is not easily feasible for a human annotator. The research community has resorted to either relying on large datasets obtained from a photorealistic simulation [8] or to smaller datasets of real-world images labeled by relying on fiduciary markers [9, 10].

### 1.2 Annotation induced Bias

We refer the reader to Section 2.1 for an overview of the datasets present in the literature. We are particularly interested in real-world datasets that rely on markers to extract the object pose ground truth. We are curious to analyze if the presence of easily recognizable shapes of markers might bring about a bias in the learning procedure, resulting in an improved success in the 6D pose estimation.

Commonly used 6D pose estimation methodologies use the Linemod (LM) dataset [9]. This dataset is labelled by fixing the target objects to a rectangular board that is surrounded by ArUco markers [11]. The ground truth object pose is then retrieved by deploying geometric algorithms which use markers to recover first the board’s pose in camera coordinates, and, successively, the object’s pose in the same coordinate system. Utilizing simple image processing algorithms and geometry, the tags are utilized to recover the board’s pose in camera coordinates, and finally the object pose is then calculated by applying the known fixed offset from the board’s coordinates system to the object’s origin.



**Figure 1** Samples from the Linemod dataset

The markers and the board used to recover the ground truth pose are visible in the training and evaluation images. Therefore, we would like to research the effect that the background and markers have on a similar model to the one detailed in Section 2.3 when predicting the 6D pose from the whole image. Additionally, the arrangement of the objects on the board could cause the model to learn a shortcut or induce unintended behavior. As Linemod is a dataset for single-object, only the pose for the object in the middle of the board is provided for training. We hypothesize that some models could leverage the aspect of other visible objects to infer the 6D pose for the target object in an unintended way.

When assessing the efficacy of a proposed method based on Linemod, or similar datasets, it is important to consider its generalization capabilities. This entails

evaluating the performance of the method when applied to different scenarios and practical applications, such as robotic manipulation or object tracking for trajectory planning purposes. Furthermore, factors such as changing backgrounds or the lack of ArUco markers or other types of markers should also be taken into account.

To this end, the following factors should be taken into account:

- How beneficial is it to have the target object positioned in the center of the board?
- To what extent is the method affected by a static or semi-static background?
- Does the network utilize 6D pose information from visible markers and the objects surrounding it?

### 1.3 Methodology overview

This paper presents a qualitative and quantitative analysis of EfficientPose (EP) [12], which was chosen as an ideal candidate for attempting to answer the aforementioned questions due to its state-of-the-art performance on Linemod [4] and its ability to operate on the full image. The purpose of this analysis is to illustrate a possible process to assess the generalization properties of a model, which is an essential requirement for any real-world application. In addition, this paper aims to emphasize the significance of selecting a proper dataset when training new models. In fact, relying on a dataset which introduces some bias could lead to deceptive outcomes.

Our work could further be situated within the field of *6D pose explainability*, an area which, to the best of our knowledge, has not been discussed previously. Given the prevalence of CNNs in computer vision and their tremendous power, it is essential to use *interpretable* models which can explain their predictions. This is important for identifying failure modes, enabling researchers to concentrate on the most promising directions. Furthermore, to ensure the reliability of CNNs in real-world applications, it is necessary to set up appropriate confidence and trust.

**Paper organization:** In Section 2, the state-of-the-art for 6D pose estimation datasets are described, insights into the 6D pose Deep Learning methods, with a focus on EfficientPose, are provided. Additionally, the Saliency maps methods for interpreting the learning process are explained. In Section 3 are described in details the experiments we have done to demonstrate our hypothesis. In Section 4 numerical and visual results are discussed, with their limitations and consequences. Finally, in Section 5 there are conclusive considerations.

## 2 Related works

### 2.1 Datasets

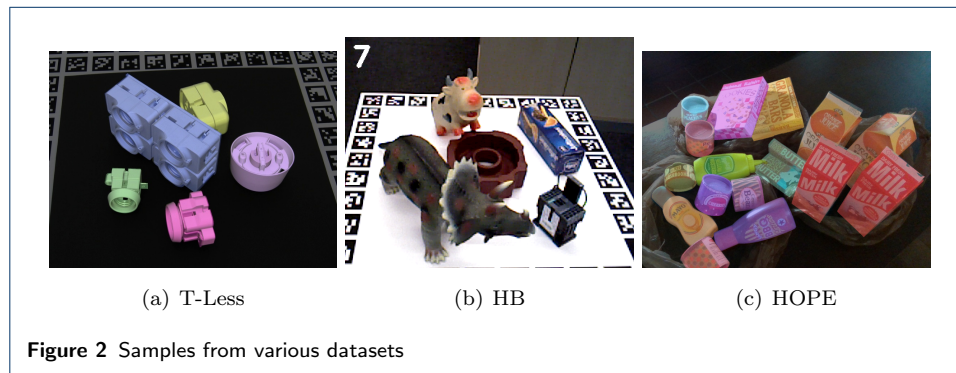
#### 2.1.1 Linemod and Linemod-Occluded

Linemod [9] is one of the most common benchmarks in the 6D pose estimation domains: it consists of real images with 15 classes (or object models), acquired from different views. A subset of images is provided for each object class, which provides ground truth 6D pose label only for the target object, which is placed around the center of a custom-made work plane and surrounded by other cluttering objects

---

[4]<https://paperswithcode.com/sota/6d-pose-estimation-on-linemod>

that cause only mild occlusion. As previously noted, the working plane consists of a chessboard-like structure delineated by custom-made ArUco markers. The work of [Linemod-Occluded \(LM-O\)](#) [10] introduces additional ground truth annotations for all modeled objects in one of the test sets, incorporating various levels of occlusion, resulting in a more challenging pose estimation task.



### 2.1.2 T-LESS

The T-LESS dataset was presented by [8], and consists of 30 industry-relevant objects which lack texture or discernible color, as well as 20 RGB-D scenes that were recorded through three synchronized cameras: the PrimeSense Carmine 1.09, the Kinect 2 RGB-D cameras, and the Canon RGB camera. The objects featured in the dataset present some symmetries and mutual similarities, with some being a combination of multiple objects.

### 2.1.3 HomebrewedDB

Structurally similar to T-LESS, [HomebrewedDB \(HB\)](#) [13] covers a wider range of objects and provides more challenging occlusions. It consists of 33 highly accurate 3D models of toys, household objects, and low-textured industrial objects of varying sizes, along with 13 sequences containing 1340 frames filmed with two RGB-D sensors. The scenes range from simple (three objects on a plain background) to complex (highly occluded with eight objects and extensive clutter). Interestingly, a chessboard-like pattern similar to the one used in Linemod is clearly visible also in HB.

### 2.1.4 HOPE

NVIDIA [Houseold Objects for Pose Estimation \(HOPE\)](#) [14] introduced a new dataset of toy grocery objects. The annotations for this dataset were obtained manually, through the identification of point correspondences between images and 3D textured object models. During the acquisition phase, ten different environments, with five object arrangements/camera poses per environment, were used. These 50 different scenes exhibit a wide variety of backgrounds, clutter, poses, and lighting. To provide additional clutter and partial occlusion, objects are also placed in other containers, such as bags or boxes. Of significance in the scope of this paper, the dataset is advantageous as it does not utilize markers or ArUco markers during acquisition. Moreover, the different environments permit to better generalize. In total,

the dataset contains 50 unique scenes, 238 images and 914 object poses. Once set the camera and the object position, some light effects are applied, in order to have more images with little differences in shadows and change colors, thereby resulting in more static images that did not need to be annotated.

## 2.2 6-DoF Pose Estimation

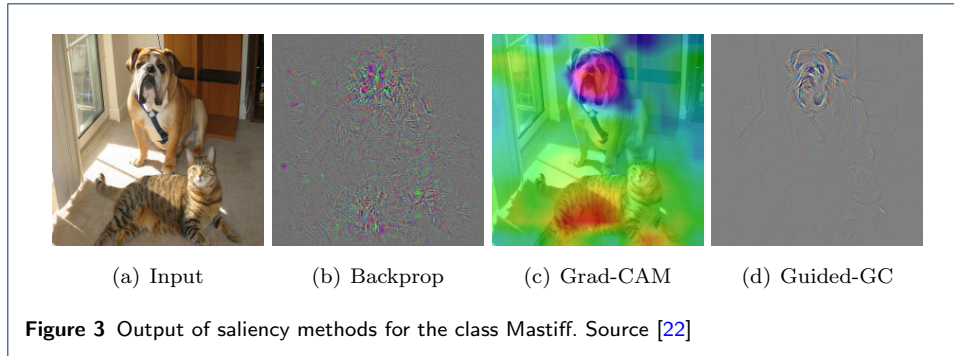
Deep-Learning based 6D pose estimation from RGB images can be divided into two approaches: *top-down*, which uses a 2D detector to identify 2D targets (either keypoints or bounding boxes) in the image before estimating the pose of each object, and *bottom-up*, which estimates the 6D pose of all the objects directly. The first category includes the keypoint-based methods which, first, extract 2D keypoints from the image, either chosen directly from the object’s surface or as 2D projections of the eight cuboid corners, and then solve a Perspective-n-Point (PnP) problem [15], [16] to recover the 6D pose. Top-down methods can also include dense methods [17], that operate by predicting the corresponding 3D model point from each 2D pixel of the object and then solving the PnP problem from sense 2D-3D correspondences between points. Bottom-up methods on the other hand in principle could simply regress the 6D pose directly from the image, however, directly estimating the 3D rotation is also difficult, since the non-linearity of the rotation space makes CNNs less generalizable. Recent works [18, 12] extend single-shoot 2D object detectors to additionally regress translation and rotation vectors for each detected object.

### 2.2.1 EfficientPose

EfficientPose [12] is an extension of a widely used 2D detector, EfficientDet (ED) [19], based on the popular convolutional backbone EfficientNet [20]. In a single shot, the architecture is able to predict the class, the 2D bounding box, rotation, and translation of one or more objects, given an RGB image as input. In detail, two analogous to the classification and bounding box regression subnetworks are added to ED, modeled after the classification and bounding-box regression of the original model. The rotation subnetwork predicts the rotation vector  $\mathbf{r} \in \mathbb{R}^3$ , in an axis-angle representation. Its architecture is similar to the class and bounding box regression, with the addition of an iterative refinement module. The final rotation is then the sum  $\mathbf{r} = \mathbf{r}_{init} + \Delta\mathbf{r}$  where  $\mathbf{r}_{init}$  is the initial estimate for the rotation, while  $\Delta\mathbf{r}$  is the iterative refinement module, given as output of separable convolutional layers, group normalization and activation functions. The translation network on the other side shares a similar structure. Instead of regressing directly  $(t_x, t_y, t_z)$ , the t-architecture predicts separately  $(c_x, c_y)$ , which represents the object center in the image, and  $t_z$ . After this,  $t_x$  and  $t_y$  are obtained from  $(c_x, c_y)$  and fixed camera parameters, as done in [21].

## 2.3 Saliency Maps

Interpretable Machine Learning (IML) [23] has experienced rapid growth in the Machine Learning domain, primarily focusing on elucidating the process behind a model’s specific prediction. While some models, such as Decision Trees and Rule Based Classifiers, are inherently interpretable, DNNs cannot be interpreted in the same manner and are generally considered to be a black-box predictor after training. Pixel attribution is a family of attribution methods designed to interpret the



prediction of models that operate on images, with the aim to produce a saliency map which highlights the importance of individual (or groups of) pixels in the input image for the model’s specific output. One of the simple methods to generate saliency maps, introduced in [24] is sometimes referred to as Vanilla-Gradient, and it consists of computing the gradient of the output prediction with respect to the input image. More formally, given  $s = \Psi(I)$ , where  $\Psi(\cdot)$  denotes the (trained) neural network, the gradient  $\frac{\partial \Psi}{\partial I}$  can be easily obtained via standard backpropagation. Grad-CAM [25] is a more recent method to produce saliency maps: differently from the vanilla gradient, the gradient of the output  $y_s$ , that is the class score before the softmax, is computed with respect to the output feature map  $A^k$  of a convolutional layer (ideally, the last one before the global average pooling of traditional classification models). The gradient tensor is then averaged across the spatial dimensions indexed by  $i, j$  to obtain a single vector  $\alpha_k^s \in R^c$  where  $c$  denotes the number of channels:

$$\alpha_k^s = \frac{1}{Z} \sum_i \sum_j \frac{\partial y_s}{\partial A_{ij}^k} \quad (1)$$

Finally, the saliency maps are obtained by weighting each channel of the feature map  $A^k$  with the corresponding value of  $\alpha_k$ :

$$L_{GC} = ReLU \left( \sum_k a_k A^k \right) \quad (2)$$

We refer the interested reader to the exceptional work of [23] for a deeper understanding of the discussed methods and several others, which falls beyond the scope of this work. In Section 3.2 we will discuss our generalization of Grad-CAM for the regression problem of our interest.

### 3 Methodology

The aim of this research is to explore the potential for bias to be introduced into the model due to the presence of visible artifacts, particularly markers employed in the data collection process. We will focus our experimentation on the Linemod dataset, a well-known non-synthetic dataset, and the EfficientPose model, which is currently the most effective fully-convolutional network for Linemod. We propose a mixed

evaluation of qualitative observations, utilizing the attribution methods discussed in Section 2.3, and quantitative experimentation, involving the evaluation of results obtained from modified versions of the Linemod dataset, to assess the validity of our hypothesis.

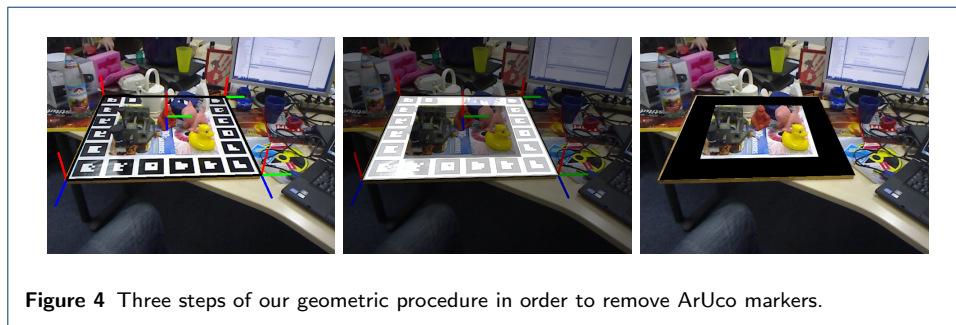
The proposed methodology, can be briefly summarized as follows:

- We propose to pre-process Linemod by deliberately masking the visible markers and use the new version of the dataset for the training of EfficientPose. The 6D pose estimation task is then compared between the late and the original training of EP.
- We also introduce a generalization of pixel attribution methods for a regression problem, and show that the saliency maps produced with the extended version of Grad-CAM support our hypothesis that the model’s predictions are contingent upon the presence of fiduciary markers on Linemod.

In the remainder of this section we detail the building blocks of the proposed methodology, then in Section 4 the conducted analysis is proposed and discussed in detail.

### 3.1 Dataset Masking strategy

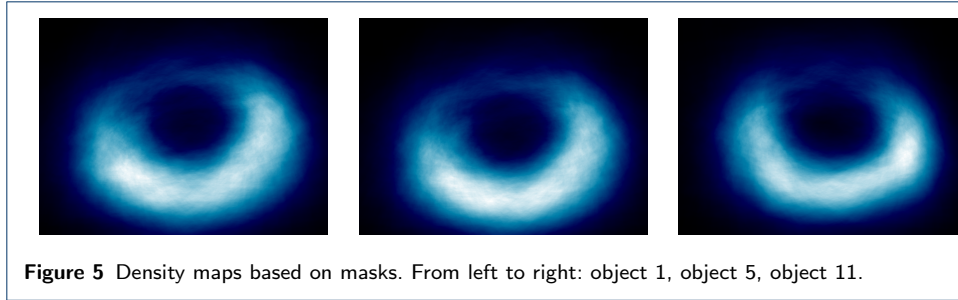
For our analysis, two datasets have been used. One is the original Linemod dataset. The other is a modified Linemod Dataset, where ArUco markers are covered with black squares. The process of deleting ArUco markers is based on geometrical tools and on the objects’ ground truth. Considering the 3D origin point  $[0, 0, 0]$  as the



**Figure 4** Three steps of our geometric procedure in order to remove ArUco markers.

object’s pivot, when no translation and rotation are applied, the positions of the black square’s four corners can be identified. The four corners correspond to the upper-left, upper-right, lower-left and lower-right corners of the ArUco chessboard. Each corner is represented by a 3D point defined as  $[\pm\delta_x, \pm\delta_y, \pm\delta_z]$ , where  $\delta_x, \delta_y, \delta_z$  are the axis offsets between the corners and the origin. Corners are identified only once for each object. Then, for each object image, the corresponding rotation and translation are applied to accurately project the black square at its correct image position (see Figure 4). AF-LM Dataset has been done in order to show and discuss how they influence final results.

In addition, ArUco masks were used also for computing a density map, able to highlight their presence. The images in Figure 5 indicate for each object which areas are ArUco markers concentrations. It is evident that these markers are not equally distributed, instead they focus on the same areas.



### 3.2 Saliency maps

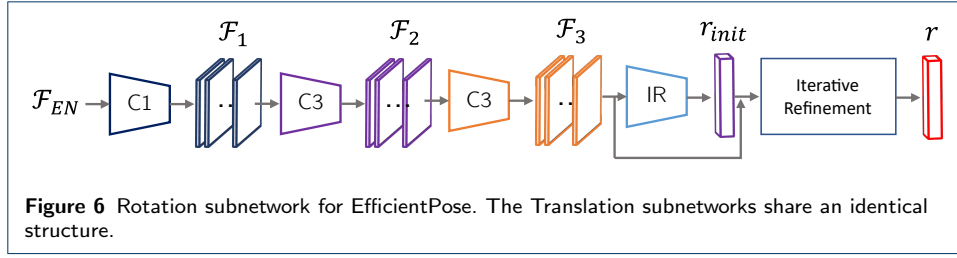
For the qualitative analysis, inspired by the domain of interpretable AI, we sought to interpret and extract information about how the network learns. This is a fundamental phase in order to assess the capabilities of 6D methods in other real-world scenarios, since metrics are limited in providing insight into a network’s comprehension.

Saliency methods are mainly developed for multi-class classification problems [26], and are thus sometimes referred to as Class Activation Maps (CAM) methods. At inference time, classification models output a probability score for each possible class, and the highest scoring class is selected as the predicted one. However, in this study, we focus on a regression problem: this is pertinent as the gradient-based methods that we plan to use, Vanilla Saliency [24] and Grad-CAM [25], suppress the negative part of the gradient since it corresponds to a decrease in the score for the class of interest. In a scenario similar to ours (i.e, regression of rotation values in  $[-\pi, \pi]$ ) we are clearly interested only in the magnitude of the gradient. Hereafter, we formalize the exact formulation of the saliency methods used in our work.

*Vanilla Saliency* It is straightforward to adapt the vanilla saliency for the regression. The Rotation head of EfficientPose outputs a tensor  $\bar{R} \in \mathbb{R}^{N \times 3}$  of  $N$  candidate regressions. In a single-object scenario, as in our case, the rotation vector  $\mathbf{r}$  for the target object is recovered as the one with the highest confidence. An identical approach can be adopted also for the translation regression. The gradient of  $\mathbf{r}$  computed with respect to the input image  $I$  is a tensor with the same shape as the input image, which is reduced to a single channel by averaging. Unlike the original formulation, we retain both the negative and positive values of the gradient for the reasons mentioned above. For visualization, the saliency map (single-channel) is normalized to the interval  $[0, 255]$  using min-max.

*Grad-CAM* Adapting Grad-CAM to our problem is far more challenging. The original formulation operates on the feature map produced by the last convolutional layer of a classification model; however, the structure of the regression head of EP (simplified in Figure 6) makes it difficult to choose the correct feature map. Therefore, we opted to use a feature map obtained as the combination of the three convolutional layers that precede the output of the initial prediction  $\bar{r}_{init}$ , as the refinement module is used to predict small additive offsets to the initial prediction, which may be Identity mappings if the initial prediction does not require refinement.





Let  $\mathcal{F}_1$ ,  $\mathcal{F}_2$  and  $\mathcal{F}_3 \in \mathbb{R}^{\overline{W} \times \overline{H} \times \overline{C}}$ , with  $\overline{W}, \overline{H}, \overline{C} \in \mathbb{N}$  be the three intermediate feature maps, we construct an aggregated featuremap  $\mathcal{F}_t \in \mathbb{R}^{\overline{W} \times \overline{H} \times 3\overline{C}} = \{\mathcal{F}_1 \oplus \mathcal{F}_2 \oplus \mathcal{F}_3\}$  where  $\oplus$  denotes the concatenation on the channel axis. Equation (1) is adapted by replacing  $A^k$  with  $\mathcal{F}_t$  and replacing the average pooling with  $L_2$  norm, in order to avoid opposing gradient values to elide each other in the sum. The new formulation for the pooled gradient becomes:

$$\alpha_t = \sqrt{\sum_{i=1}^{\overline{W}} \sum_{j=1}^{\overline{H}} \left( \frac{\partial \mathbf{r}}{\partial \mathcal{F}_t^{ij}} \right)^2} \in \mathbb{R}^{3\overline{C}}$$

The computation of the saliency maps is obtained by weighting each channel of  $\mathcal{F}_\square$  with the corresponding value of  $\alpha_k$  and accumulating over the channel axis to obtain a single matrix. Differently from eq. (2) the ReLU is removed and the absolute value of the weighed feature map is instead taken:

$$L_{GC} = \sum_{t=1}^{3\overline{C}} |a_t \mathcal{F}_t|$$

The produced saliency map is normalized and interpolated to the input image shape for visualization. The formulation, derived from simple but solid mathematical observations, results in an ideal generalization of Grad-CAM to our case.

### 3.3 Evaluated Task metrics

The metrics that we used to test the networks are two:

- **ADD**: the average distance computes the mean distance between each point of the 3D model obtained by the pose matrices  $\hat{\mathbf{P}} = [\mathbf{R}_{\text{est}}, \mathbf{t}_{\text{est}}; \mathbf{0}, 1]$  and  $\overline{\mathbf{P}} = [\mathbf{R}_{\text{gt}}, \mathbf{t}_{\text{gt}}; \mathbf{0}, 1]$

Given the model  $\mathcal{M}$ , the estimated pose  $\hat{\mathbf{P}}$  and the ground truth  $\overline{\mathbf{P}}$

$$e_{ADD} = \text{avg}_{x \in \mathcal{M}} \|\hat{\mathbf{P}}\mathbf{x} - \overline{\mathbf{P}}\mathbf{x}\|$$

- **ADD-S**: the average closest point distance computes the mean distance between each point of the 3D estimated model and its closest neighbor on the ground truth model:

$$e_{ADD-S} = \text{avg}_{\mathbf{x}_1 \in \mathcal{M}} \min_{\mathbf{x}_2 \in \mathcal{M}} \|\hat{\mathbf{P}}\mathbf{x}_1 - \overline{\mathbf{P}}\mathbf{x}_2\|$$

It is preferred if the model  $\mathcal{M}$  has indistinguishable views.

- **Criterion of Correctness** The estimated pose is considered correct if  $e < \theta_{AD} = k_m d$  where  $k_m$  constant generally equal to 0.1,  $d$  = object diameter

### 3.4 Proposed Evaluations

For our experimental analysis we focus on the Rotation subtask of **EP**, since the translation regression is based on an identical structure, as introduced in section 2.2.1, the same principles are agnostic to the regression subtask. We compare the results, both qualitatively and quantitatively, obtained by two architecturally identical instances of **EP**: the original model trained on the official Linemod dataset and an alternative version trained on the ArUco-Free dataset introduced in Section 3.1. Since we focus on single-class **EP**, the evaluation procedure is performed independently for three distinct representative object classes from Linemod.

To summarize, a total of 6 distinct versions of **EP** are trained: we pick the standard version of **EP** ( $\phi = 0$ ) and train on subsets of objects 1 (Ape), 5 (Can) and 11 (Glue) of both original (**LM**) and **Aruco-Free Linemod (AF-LM)** datasets. For each trained model, we compute **6D**-pose metrics (section 3.3) and saliency maps (section 3.2) on both the validation subsets of **LM** and **AF-LM** for the corresponding object. In the following section we provide the complete analysis of the proposed study.

## 4 Results and Discussion

### 4.1 Quantitative analysis

We considered three objects: one with symmetric views (object 11, glue) and the other two asymmetric (object 1, ape and object 5, can). Considering **ADD(-S)** as a metric, the following rule is used:

$$ADD(-S) = \begin{cases} ADD - S, & \text{if } obj \text{ sym} \\ ADD, & \text{if } obj \text{ asym} \end{cases}$$

#### 4.1.1 Object 1: the ape

Table 1 shows how EfficientPose performs on the two different datasets. We can observe that in both training (with and without ArUco markers) the network learns from the background. In fact, if the test dataset is different from the train one, the pose estimation accuracy collapses. Figure 7 compares estimated (blue) with ground truth (green) bounding boxes, with weights learned on the Original **LM** Dataset. When ArUco-Free Dataset is used as test, in some cases the rotation is wrong, in others the object is not even detected. Probably, when the ArUco markers are covered, the network still learns from the black square around the object. Object 1 achieves the worst performances on both datasets.

#### 4.1.2 Object 5: the can

For object 5, the can, we used trained EfficientPose weights with  $\phi = 0$  on the Original **LM** Dataset and on the ArUco-Free Dataset. The object is not symmetric, therefore **ADD** is used. For this object, we noticed that the results are similar to the ones of object 1, and confirm our thesis of background-induced bias. However, in this case, EfficientPose with our ArUco-Free Dataset performs better in both cases (Original **LM** and **AF-LM** datasets), as shown in Table 2.



#### 4.1.3 Object 11: the glue

The third object we chose for our study is a symmetric object, for this reason, we show ADD-S results in Table 3. They are higher since ADD-S has more relaxed constraints than ADD (as can be deduced by the definition). In this case, while performances on the same dataset of the training are almost perfect, with an accuracy of 100%, the training without ArUco performs better than the other with the opposite dataset. Therefore, the average accuracy is better. From these metrics it is evident that the network learns from the background. However, accuracies don't give us information about which background areas are more relevant than others. To explain better these results, we used saliency maps in the next section.

#### 4.2 Qualitative analysis

Figure 8 shows Vanilla Saliency maps. They are computed with respect to two Original LM images. The weights are given by EfficientPose code for object 1, with  $\phi = 0$ . The map represents the gradient magnitude for each pixel. Since EP has a first common feature extraction phase and, then, is divided into different subnetworks with their own output (classification, bounding boxes, rotation, and

Test			
Original LM training	Original LM 0.8771	AF-LM <b>0.0162</b>	<b>0.4467</b>
AF-LM training	AF-LM <b>0.8848</b>	Original LM 0.0	0.4424

**Table 1** This is object 1. ADD is computed on the two datasets with the two types of weights available (trained with  $\phi = 0$ ). Then, in the right column, for each training, the two test results' averages are computed, in order to observe which experiment performs better both on the test images of the same image type seen during training, and on the test images of the type never seen.

Test			
Original LM training	Original LM 0.9852	AF-LM 0.0315	0.5083
AF-LM training	AF-LM <b>0.9921</b>	Original LM <b>0.1673</b>	<b>0.5797</b>

**Table 2** This is object 5. ADD is computed on the two datasets with the two types of weights available (trained with  $\phi = 0$ ). Then, in the right column, for each training, the averages of the two test results are computed, in order to observe which experiment performs better both on the test images of the same type of the images seen during training, and on the test images of the type never seen.

translation), two saliencies for each image have been compared. The 'rotation' image represents vanilla saliency map based on the rotation subnetwork, while the 'classification' image comes from the classification subnetwork. The differences for object 1 in this image are significant. In fact, while the saliency for classification is grouped into the principal object, the ape, instead the rotation saliency is more scattered and goes also on the marker chessboard. It probably means what we expected: that, for the position output, a bias is induced by background, which plays a fundamental role. In order to improve our study with a more sophisticated and explainable saliency method, we also computed saliency maps with GradCAM method. Moreover, sometimes Vanilla Saliency could have a saturation problem, as shown by [25] and [23].

In Figure 9, we can observe different results for the three objects. The weights are learned on the Original LM, while the tests are computed on both datasets. For example, for object 1 (on the left), pixels with higher saliency values are, on the Original LM (top), distributed on the object and the markers, whereas on the AF-LM, they do not put focus on markers, in fact they are covered. It means that, when ArUco values collapse to zero, they are not anymore interesting for the rotation estimation.

For object 5, in the center, the focus is on a large area which includes also the can. Therefore, the network uses not only the principal object, but also the background one, to estimate the final pose. This is possible since background objects do not change their position with respect to the can.

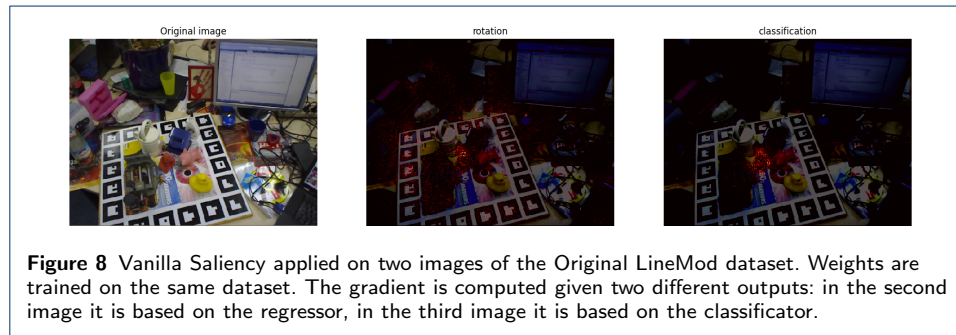
Similar to the ape behavior, for object 11's pose estimation EP focuses on marker chessboard, not even observing the principal object. When markers are zeroed, saliency is distributed in not well-defined areas. These images prove the fact that there is a background-bias during EfficientPose training with ArUco markers.

Figure 10 represents saliency maps obtained from our weights, learned on the AF-LM. These results are quite different from the previous ones.

For object 1 the saliency is not on the markers, however, the network doesn't focus on the ape, looking for information in other background objects. Saliency does not change a lot: it means that markers are not the central keypoints for pose estimation.

Test			
Original LM training	Original LM <b>1.0000</b>	AF-LM 0.3031	0.6516
AF-LM training	AF-LM <b>1.0000</b>	Original LM <b>0.4083</b>	<b>0.7042</b>

**Table 3** This is object 11. ADD-S is computed on the two datasets with the two types of weights available (trained with  $\phi = 0$ ). Then, in the right column, for each training, the two test results' averages are computed, in order to observe which experiment performs better both on the test images of the same image type seen during training, and on the test images of the type never seen.



Nevertheless, the network on the Original LM dataset has the worst performance accuracy.

Also, in object 5 (in the center) images saliency maps seem to remain the same for both tests.

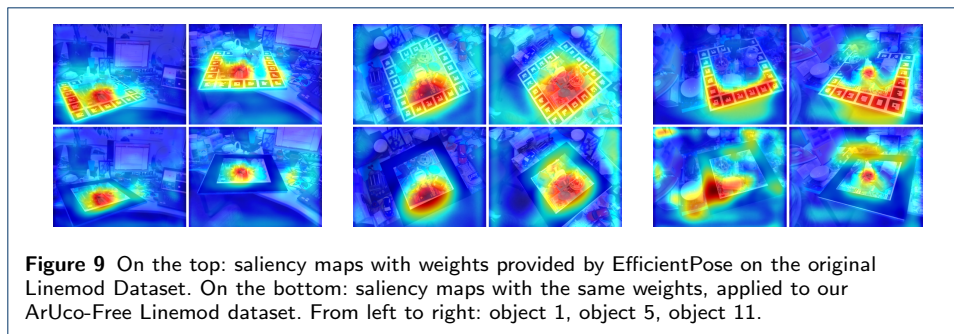
An interesting phenomenon happens with object 11 (on the right). The most plausible interpretation is that weights on the ArUco-Free dataset predict the pose based on the square edges. When these edges are not so strongly highlighted due to the presence of ArUco markers, the prediction is wrong. For this reason, we think that covering the ArUco markers is useful for uncovering the bias, but it's not enough for obtaining a more generalized dataset.

An interesting result emerges from the saliency maps calculated for the model trained on the Original LM dataset. For the original Linemod images, the saliency maps focus on both the ArUco markers and the area close to the target object, as described above and depicted in Figure 9 (top). In contrast, for the ArUco-Free Linemod images, the saliency maps are sometimes zeroed out. This observation can be explained by the fact that, for ArUco-Free Linemod images, some pixels were zeroed for each RGB channel. The zeroing of these pixels, which refer to the ArUco markers considered essential for the EP model, causes the gradients to be zeroed as well. It is as if these null values made that part of the hyperplane on which the gradients were calculated constant.

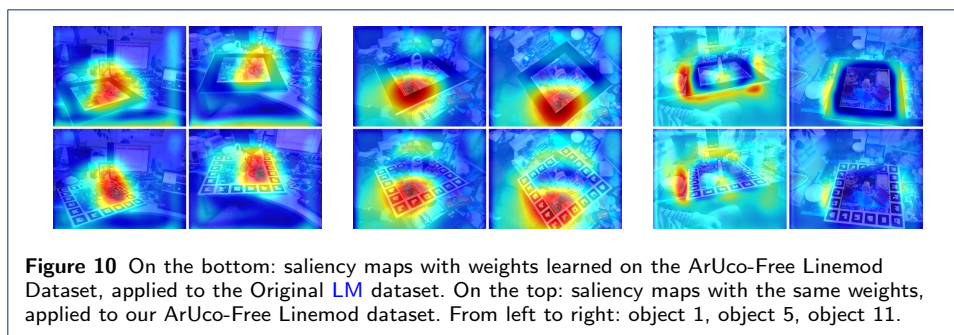
Indeed, the gradient is zero for any constant value, as well as for a minimum or maximum point. Furthermore, for these specific images, we verified that the last activations before the last convolutional layer used for the Grad-CAM calculation were not null.

### 4.3 Consequences

Finally, this paper aims to give some advices for avoiding background-induced bias in 6D object pose estimation. Foremost, networks are preferable which first detect the object, and then predict the pose of the cropped image. Secondly, the dataset



**Figure 9** On the top: saliency maps with weights provided by EfficientPose on the original Linemod Dataset. On the bottom: saliency maps with the same weights, applied to our ArUco-Free Linemod dataset. From left to right: object 1, object 5, object 11.



**Figure 10** On the bottom: saliency maps with weights learned on the ArUco-Free Linemod Dataset, applied to the Original LM dataset. On the top: saliency maps with the same weights, applied to our ArUco-Free Linemod dataset. From left to right: object 1, object 5, object 11.

choice is also fundamental. To avoid biasing the learning method, it is important to change the background from training frames. In particular, different lights, views and background objects are helpful to overcome this issue. For example, HOPE [14], as written in section 2.1, presents images in different scenarios, changing also the lights. In addition, we propose to change the markers to avoid repetitive appearances across consecutive frames.

#### 4.4 Future Purposes

This paper is limited to one dataset and focuses on EfficientPose, but our future purposes include working on different existing 6D methods, in order to compare networks not only for their final ADD(-S) metric, but also for their generalization skills and their applicability. Moreover, we plan to release the ArUco-Free Dataset, such that everyone could use it and measure generalization skills of their own model.

## 5 Conclusions

We analyzed the effect of the presence of ArUco markers in the dataset regarding the learning procedure. In particular we found that they induce bias in the performance of the methodology. We assessed the generalization capabilities of one of the state-of-the-art 6D pose estimation networks, with respect to the two different datasets: with and without ArUco markers. The outcome was that the model trained on the dataset without markers achieves better results. However, we substituted ArUco markers with a black zone, and the black squares instead of markers are not enough to generalize. To achieve a better generalization, the dataset should be augmented, with different lighting conditions, different views and more varied backgrounds. Our ArUco-Free LineMod dataset could be useful to prove more generalization capabilities of any newly proposed method. In addition, the object's position, always

in the center of the board is not beneficial from the generalization point of view, and it's preferable to crop the object once it's been detected.

Furthermore, the saliency analysis showed that the network definitely utilizes 6D pose information from visible markers and background objects.

This paper is the starting point to investigate the potential presence of bias in DNNs trained for 6D pose estimation, and to propose new methods to solve this problem.

## Abbreviations

**6D** 6-Degrees-of-Freedom. 1–3, 5, 7, 8, 10, 13–15

**AF-LM** Aruco-Free Linemod. 7, 10, 12, 13

**CNNs** Convolutional Neural Networks. 1, 3, 5

**DNNs** Deep Neural Networks. 1, 5, 15

**EP** EfficientPose. 3, 7, 8, 10–13

**HB** HomebrewedDB. 4

**HOPE** Houseold Objects for Pose Estimation. 4

**LM** Linemod. 2, 3, 10–14, 17

**LM-O** Linemod-Occluded. 4

## Availability of data and material

[GitHub Repository](#)

## Competing interests

The authors declare that they have no competing interests.

## Funding

This research received no external funding.

## Author's contributions

All authors, EG and DS and CS and TP and GF and PA and MV and MB, contributed equally to the study conception and design. All authors, EG and DS and CS and TP and GF and PA and MV and MB, have read and agreed to the published version of the manuscript.

## Acknowledgements

This work was also supported by the “Gruppo Nazionale per il Calcolo Scientifico (GNCS-INdAM)”.

The publication was created with the co-financing of the European Union-FSE-REACT-EU, PON Research and Innovation 2014-2020 DM1062/2021.

### Author details

<sup>1</sup>Department of Physics, Informatics and Mathematics, University of Modena and Reggio Emilia, Modena, IT.

<sup>2</sup>Department of Mathematical, Physical and Computer Sciences, University of Parma, Parma, IT. <sup>3</sup>TII Technology Innovation Institute, Abu Dhabi, UAE. <sup>4</sup>Hipert SRL, Modena, IT.

## References

1. Wuest, H., Stricker, D.: Tracking of industrial objects by using cad models. *JVRB-Journal of Virtual Reality and Broadcasting* **4**(1) (2007)
2. Zhu, M., Derpanis, K.G., Yang, Y., Brahmabhatt, S., Zhang, M., Phillips, C., Lecce, M., Daniilidis, K.: Single image 3d object detection and pose estimation for grasping. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 3936–3943 (2014). IEEE
3. Sapienza, D., Govi, E., Aldaheri, S., Franchini, G., Bertogna, M., Roura, E., Èric Pairet, Verucchi, M., Ardón, P.: Model-Based Underwater 6D Pose Estimation from RGB (2023). [2302.06821](#)
4. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1907–1915 (2017)
5. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3354–3361 (2012). IEEE
6. Marchand, E., Uchiyama, H., Spindler, F.: Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics* **22**(12), 2633–2651 (2015)
7. Shapiro, L.G., Stockman, G.C., et al.: *Computer Vision* vol. 3. Prentice Hall New Jersey, ??? (2001)
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: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 880–888 (2017). IEEE
9. 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, pp. 548–562 (2012). Springer
10. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6d object pose estimation using 3d object coordinates. In: European Conference on Computer Vision, pp. 536–551 (2014). Springer
11. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F.J., Marín-Jiménez, M.J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition* **47**(6), 2280–2292 (2014)
12. Bukschat, Y., Vetter, M.: Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach. *arXiv preprint arXiv:2011.04307* (2020)
13. Kaskman, R., Zakharov, S., Shugurov, I., Ilic, S.: Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, pp. 0–0 (2019)
14. Tyree, S., Tremblay, J., To, T., Cheng, J., Mosier, T., Smith, J., Birchfield, S.: 6-dof pose estimation of household objects for robotic manipulation: An accessible dataset and benchmark. *arXiv preprint arXiv:2203.05701* (2022)
15. Li, S., Xu, C., Xie, M.: A robust o (n) solution to the perspective-n-point problem. *IEEE transactions on pattern analysis and machine intelligence* **34**(7), 1444–1450 (2012)
16. Lepetit, V., Moreno-Noguer, F., Fua, P.: Eppn: An accurate o (n) solution to the pnp problem. *International journal of computer vision* **81**(2), 155–166 (2009)
17. Li, Z., Wang, G., Ji, X.: Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 7678–7687 (2019)
18. Kehl, W., Manhardt, F., Tombari, F., Ilic, S., Navab, N.: Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1521–1529 (2017)
19. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10781–10790 (2020)
20. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, pp. 6105–6114 (2019). PMLR
21. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199* (2017)
22. Ozbek, U.: PyTorch CNN Visualizations. GitHub (2019)
23. Molnar, C.: A guide for making black box models explainable. URL: <https://christophm.github.io/interpretable-ml-book> (2018)
24. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013)
25. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 618–626 (2017)
26. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. *International journal of computer vision* **115**(3), 211–252 (2015)

## List of Figures

- |   |  |   |
|---|--|---|
| 1 | Samples from the Linemod dataset . . . . .                                       | 2 |
| 2 | Samples from various datasets . . . . .  | 4 |
| 3 | Output of saliency methods for the class Mastiff. Source [22] . . . . .          | 6 |
| 4 | Three steps of our geometric procedure in order to remove ArUco markers. . . . . | 7 |



5 Density maps based on masks. From left to right: object 1, object 5, object 11. . . . . 8

6 Rotation subnetwork for EfficientPose. The Translation subnetworks share an identical structure. . . . . 9

7 These images are used during the test phase on object 1 with weights downloaded from the available Efficient Pose training ( $\phi = 0$ ). The green box represents the ground truth pose, while the blue one represents the estimated pose. On the first and third rows there are test images from the original Linemod dataset, while on the second and fourth rows there are the correspondent images from ArUco-Free dataset. We observed that the boxes are all wrong and, in some cases, the network doesn't even detect the object. . . . . 11

8 Vanilla Saliency applied on two images of the Original LineMod dataset. Weights are trained on the same dataset. The gradient is computed given two different outputs: in the second image it is based on the regressor, in the third image it is based on the classifier. . . 13

9 On the top: saliency maps with weights provided by EfficientPose on the original Linemod Dataset. On the bottom: saliency maps with the same weights, applied to our ArUco-Free Linemod dataset. From left to right: object 1, object 5, object 11. . . . . 14

10 On the bottom: saliency maps with weights learned on the ArUco-Free Linemod Dataset, applied to the Original LM dataset. On the top: saliency maps with the same weights, applied to our ArUco-Free Linemod dataset. From left to right: object 1, object 5, object 11. . . 14