YOLOff: You Only Learn Offsets for robust 6DoF object pose estimation

Mathieu Gonzalez[1], Amine Kacete[1], Albert Murienne[1], Eric Marchand[2]

[1]Institute of Research and Technology b<>com
[2] Univ Rennes, Inria, IRISA, CNRS, Rennes, France

Abstract. Estimating the 3D translation and orientation of an object is a challenging task that can be considered within augmented reality or robotic applications. In this paper, we propose a novel approach to perform 6 DoF object pose estimation from a single RGB-D image in cluttered scenes. We adopt an hybrid pipeline in two stages: data-driven and geometric respectively. The first data-driven step consists of a classification CNN to estimate the object 2D location in the image from local patches, followed by a regression CNN trained to predict the 3D location of a set of keypoints in the camera coordinate system. We robustly perform local voting to recover the location of each keypoint in the camera coordinate system. To extract the pose information, the geometric step consists in aligning the 3D points in the camera coordinate system with the corresponding 3D points in world coordinate system by minimizing a registration error, thus computing the pose. Our experiments on the standard dataset LineMod show that our approach is more robust and accurate than state-of-the-art methods.

Keywords: 6D object pose estimation · Patch based · 3D registration
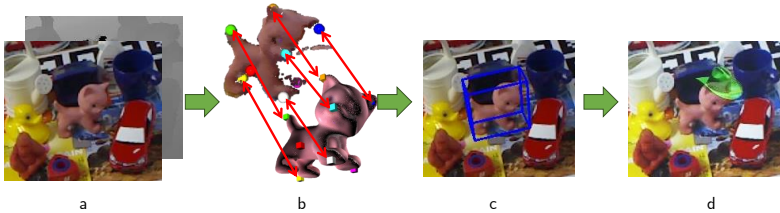
1 Introduction

Fig. 1: To estimate the pose of an object we propose to use a single RGB-D image (a) to predict the position of a set of sparse 3D keypoints (shown as spheres in b) in the camera coordinate system, which are then registered in 3D with corresponding points in the world coordinate system (shown as cubes in b) to retrieve the pose (c) that can be used to insert a virtual object (d) in the scene as an AR application.

The goal of object pose estimation is to predict the rotation and position of an object relative to a known coordinate frame (usually the camera coordinate

frame). This computer vision problem has many applications such as augmented reality or robotics. In the former case it allows a realistic insertion of virtual objects in an image as described in [1] and shown in figure 1. In the latter it can be used as an input for a robotic arm to grasp and manipulate the object such as in [2]. Although heavily studied, this problem is still relevant as it is unresolved due to its complexity. Indeed some scenes can be highly challenging due to the presence of cluttering, occlusions, changes in illumination, viewpoint, and textureless objects. Nowadays color and depth (RGB-D) sensors are smaller and cheaper than ever, making them relevant for object pose estimation. Indeed, compared to color-only (RGB) sensors, the depth channel provides relevant information for estimating the pose of textureless objects in dimly lit environments.

Classical object pose estimation approaches are either based on local descriptors followed by 2D-3D correspondences [1], or on template matching [3,4,5]. However the challenging cases listed above limit their performance. To address these limitations most recent methods solve the problem of object pose estimation with a data driven strategy using for example Convolutional Neural Networks (CNNs) [2,6,7,8,9,10,11,12,13]. These approaches work in a holistic way, considering a whole RGB or RGB-D image as an input and making a single estimation of the pose. While some methods are hybrid, using a learning-based approach followed by a geometrical solver [6,7,8,9,10], others use an end-to-end CNN to predict the pose [2,11,12].

Some older methods however have proven to be reliable using patch voting approaches coupled to a learning algorithm [3,5,14,15,16]. Those strategies predict a set of pose hypothesis from local patches using data driven functions and, from this set of hypothesis, retrieve a stronger, more robust pose.

We argue that we can leverage the robustness brought by local approaches with a two stages strategy, predicting the pose in an intermediate Euclidean 3D space and retrieving it with a geometrical solver. The intermediate representation makes it natural to apply a voting strategy to the set of pose hypothesis. Our hybrid strategy allows us to correctly supervise our CNN training, not being dependent on the choice of pose representation, not requiring a custom loss function to compute the pose error and not having to predict rotation and translation separately.

In this paper we tackle the problem of pose estimation considering a single RGB-D image as input. We design a robust and accurate algorithm to predict the pose of a generic rigid object in a scene. Our contributions are :

- We propose an hybrid pipeline in two parts: a data driven block that predicts a set of 3D points in the camera coordinate system and a geometrical block. The latter retrieves the pose given the estimated points and a priori chosen keypoints in the world coordinate system, minimizing a registration error.
- We propose to use two CNNs in cascade in the former part. First we predict the object 2D location in the image, classifying local patches extracted from the image with a CNN. Then we use a regression CNN to predict a set of possible 3D positions of points in the camera coordinate system. The position

hypothesis are then robustly aggregated to obtain a single estimation of the points 3D location.
- We demonstrate performance improvements in terms of accuracy over state-of-the-art methods of RGB-D pose estimation on the standard LineMod [4] dataset.

2   Close work

We will limit ourselves to learning based methods as the literature on object pose estimation is vast. We can separate those methods into two main categories: patch-based methods and holistic methods. The latter can be as well separated into two categories: direct and indirect strategies.

Patch-based Methods. Patch-based methods output multiple pose hypothesis for a single image [3,5,14,15,17]. The predictions, called votes, which are obtained from local patches in the image are then aggregated to get a single estimation, which is more robust than each vote taken independently. Hough based methods is such a type of voting scheme. Hough Random Forests (HRFs) have been introduced by [17] to estimate the Hough transform with a learning based approach for object detection, tracking in 2D and actions recognition. The concept of HRFs has also been applied to object pose estimation by [15] to predict the translation and rotation of human heads. In that case, both the nose 3D position and Euler angles are regressed. Those methods rely on binary tests to describe the split hypothesis used in random forests. [3] proposes to use a split function based on a template patch descriptor. It also proposes to train a random forest using only object patches. As HRFs are based on handcrafted split functions, their performance is limited by image variations. To overcome this, Hough Convolutional Neural Networks (HCNNs) have been introduced by [14] as an alternative to HRFs. A CNN was designed by [14] to regress at once the probability of a patch belonging to the foreground as well as the object pose. In all cases a non parametric clustering algorithm is then used on object patches to robustly retrieve the pose.

Direct Holistic Methods. Recently, most studies [2,11,12,18,19] take a whole image as an input and try to leverage the capabilities of CNNs by directly estimating the pose. PoseCNN [12] proposes an end-to-end CNN to perform 3 related tasks: semantic labeling, translation prediction from the object estimated 2D center and depth and rotation inference. To correctly supervise the network training, [12] uses a specific loss called PoseLoss, defining the error as an average euclidean distance between rotated point clouds. SSD6D, [19], uses a CNN to predict the object class with its bounding box, as well as to classify discretized viewpoints and in-plane rotations to create a set of pose hypothesis. Thus, the network loss is a parametric combination of multiple losses. [11] proposes an analysis-by-synthesis approach, iteratively rendering the object with an estimated pose to refine it. A change of coordinate system allows to regress rotations and translations separately. DenseFusion, [2], combines color and depth

channels in a deep network to fuse them, creating a set of features which are then used by a CNN to predict the pose. It can be further rapidly refined by a network in an iterative manner. In some recent works [18,20,21] the choice of representation for rotations has been studied as it shows to have an impact on the accuracy of the pose estimation [21]. To remove the constraints imposed by some representations and get a minimal number degrees of freedom [18] proposed to represent rotations using Lie algebra elements.

Indirect Holistic Methods. On the other side some methods [6,7,8,9,22] are inspired by classical pose estimation from 2D-3D correspondence. However CNNs are used to address the limits imposed by handcrafted features. To do so the 2D location of the projection of prior chosen 3D keypoints is predicted in the image. The pose is then retrieved using a 2D-3D geometrical solver e.g. a Perspective-n-Points (PnP) algorithm. For example BB8, [6] coarsely segments the object and apply a deep convolutional network to the local window around the object to predict the 2D location of the projection of the 8 corners of the object bounding box. This estimation is followed by a PnP that recovers the pose. [8] proposes a single-shot CNN that classifies the object, predicts a confidence term as well as the 2D location of the projection of 9 keypoints in the bounding box. [22] uses a stacked hourglass architecture to produce a 2D heatmap of class specific keypoints, an optimization problem is then solved to retrieve the pose. PVNet[7] proposes to apply an offset based approach to predict the 2D location of a set of keypoints on the object surface. To do so, they segment the object in the image and predict a vector field on the segmented object, the spatial probability distribution of each keypoint is then retrieved and used in an uncertainty driven PnP to estimate the pose. H+O [13] estimates at once hand-object poses as well as objects and actions classes from RGB images. A CNN predicts the 3D position of 21 points of the object bounding box and the object pose can be retrieved from 3D-3D correspondences.

3   Proposed approach: our hybrid, patch-based strategy

Our goal is to achieve a robust and accurate 6-DoF pose estimation of a 3D object, i.e. to estimate the transformation $\mathbf{T} \in SE(3)$ of an object of interest, provided with a coordinate system called world coordinate system $\mathcal{F}_w$, in the camera coordinate system $\mathcal{F}_c$. We represent the transformation $\mathbf{T}$ between $\mathcal{F}_c$ and $\mathcal{F}_w$ as

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} \ \mathbf{t} \\ \mathbf{0} \ 1 \end{pmatrix} \tag{1}$$

where $\mathbf{R} \in SO(3)$ is a 3D rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is a 3D translation vector.

The pipeline of the proposed strategy can be seen figure 2: we adopt a patch voting based approach inspired from [14,17], using multiple local information to predict a sparsified version of the object geometry in $\mathcal{F}_c$. First, we design and train a classification CNN to predict the class of patches extracted from the input image either as object or background. This allows us to roughly localize

Fig. 2: Overview of our pipeline solution: first (a), patches are extracted from an RGB-D image and classified as object or background. Next (b), for each object patch, a regression CNN predicts a set of vectors estimating the position of specific 3D keypoints in $\mathcal{F}_c$. Those votes are then aggregated in a non parametric way to obtain a robust estimation of the points in $\mathcal{F}_c$. (c) by minimizing the registration error between corresponding estimated keypoints in $\mathcal{F}_c$ and reference keypoints in $\mathcal{F}_w$ we retrieve the 6D pose.

the object in 2D. Then, we design and train a regression network to predict for each extracted object patch the 3D position in $\mathcal{F}_c$ of a set of prior chosen keypoints, selected in $\mathcal{F}_w$. Finally, by minimizing the 3D-3D registration error between corresponding estimated keypoints in $\mathcal{F}_c$ and reference keypoints in $\mathcal{F}_w$ we retrieve the 6D pose.

3.1   2D Localization

In this section we show how we take in account the visibility of the object in each patch. Indeed not all patches contain relevant information about the object pose. Unlike in [14] who uses a single network for both classification and pose estimation, we first use a classification network to decide whether or not a patch contains a representation of the object. We argue that classifying the patches, keeping only relevant ones before transmitting them to the regression network allows the CNN to fit using only relevant information about the object pose. Moreover we do not need a sophisticated parametric loss function whose parameters have to be optimized to supervise the training.

Model. Our model is inspired by a light VGG-like architecture and can be seen in the first block of figure 2. It is composed of a set of convolutional layers to extract features from the images and max-pooling layers to introduce scale invariance followed by 2 dense layers to classify the extracted features. For the last layer, we use a sigmoid activation function, for each other layer we use the

classical ReLu activation function. We used less neurons and layers than the original VGG architecture both to reduce overfitting and because the classified patches are small. To help reduce overfitting dropout is also used on the first fully connected layer as it contains the most weights.

Data. To train our classification network in a supervised manner, we need labeled data. We capture a set of images representing the object of interest from multiple points of view. The classification neural network is trained using a set of patches $\{P_i = (\mathbf{I}_i, b_i)\}$ where $\mathbf{I}_i$ is the RGB image of the patch of size $[h \times w]$, i.e. $\mathbf{I}_i \in \mathbb{R}^{[h \times w] \times 3}$ and $b_i \in [0, 1]$ represents whether or not the object is visible in the image $\mathbf{I}_i$. We obtain it by producing a binary mask of the object created by a 2D projection of the object 3D model using its ground truth pose. To increase the robustness of our algorithm across changes in illumination we proceed to do data augmentation by randomly modifying patches brightness.

Training. We denote the classification function f_{θ_c} optimized over θ_c which represents our CNN weights. The classification parameters are optimized by minimizing over the training data set:

$$\theta_c^* = \arg\min_{\theta_c} \mathcal{L}_c(b, \hat{b}). \tag{2}$$

where $\hat{b} = f_{\theta_c}(\mathbf{I})$ and $\mathcal{L}_c$ is the weighted binary cross entropy:

$$\mathcal{L}_c(b, \hat{b}) = -(w_0 b \log(\hat{b}) + (1 - w_0)(1 - b)\log(1 - \hat{b})) \tag{3}$$

with b, the binary indicator corresponding to the class label, $\hat{b}$ the predicted probability and w_0 the weight given to the class 0.

Inference. Given an unseen image, we densely extract K patches from the image and get a set of patches $\mathcal{P} = \{P_i, i \in [1, K]\}$. Each patch is then fed to the classification network $f_{\theta_c^*}$ whose output is $\left\{\hat{b}_i = p(b_i | \mathbf{I}_i) = f_{\theta_c^*}(\mathbf{I}_i), i \in [1, K]\right\}$ where p denotes the probability. We show in figure 3 some heat maps obtained using the probability estimated for each patch. We can see that the patches extracted from the object have a high probability of being classified as object while the patches extracted from the background have a low probability, except for a few patches that are misclassified due to their local similarity with the object.

3.2   3D points prediction

We now show how we predict the position of a set of 3D keypoints in $\mathcal{F}_c$, using the object patches classified in the previous step. We use a regression network to predict the 3D location of a set of M points in $\mathcal{F}_c$. First, we create a set of M 3D keypoints, denoted $\mathcal{S} = \{\mathbf{X}_j \in \mathbb{R}^3, j \in [1, M]\}$, chosen in the object model in $\mathcal{F}_w$. For a given pose $\mathbf{T}$ of the object in $\mathcal{F}_c$ we express the points in $\mathcal{S}$ in $\mathcal{F}_c$,
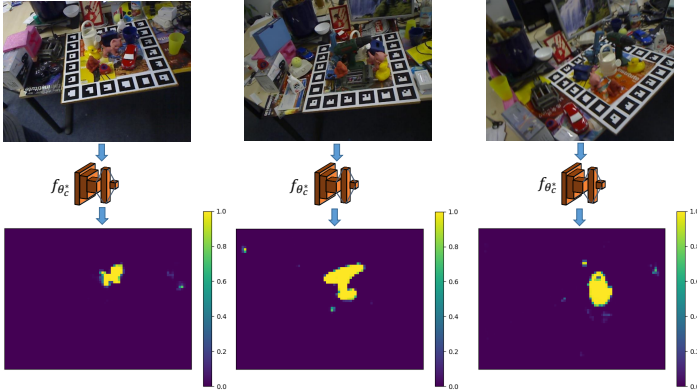
Fig. 3: Examples of probability maps for the cat, driller and can from the LineMod dataset.

and denote the set $\mathcal{S}_{\mathbf{T}} := \{\mathbf{Y}_j := \mathbf{T}\mathbf{X}_j, j \in [1, M]\}$. Our goal is to estimate the location of the points in $\mathcal{S}_{\mathbf{T}}$ i.e. to estimate the location of the keypoints of $\mathcal{S}$ in $\mathcal{F}_c$. We argue that it is easier for the neural network to predict points in a euclidean space than to predict a pose over $SE(3)$. Let us recall that no distance exists over $SE(3)$ which makes a loss function very difficult to exhibit. Like [9] we argue that rotations and translations should be treated differently or at least that adaptation is required to learn to regress coherently in $SE(3)$. In a way with our change of variables we suppress the direct impact of the peculiarities of rotation space as every variable stays in $\mathbb{R}^3$.

Model. The architecture of the regression network can be seen in the second block of the figure 2. We use an architecture that is very close to the classification network because we showed that we could reliably extract information from the patches with it. However we change the fully connected part, adding one layer and using more weights for each layer to give the regression CNN more flexibility.

Data. We extract only object patches P'_i from the image. A regression neural network is trained using a set of patches $\mathcal{P}' = \{P'_i = (\mathbf{I}'_i, \boldsymbol{\delta}_i)\}$ where $\mathbf{I}'_i$ is the RGB-D image of the patch, i.e., $\mathbf{I}'_i \in \mathbb{R}^{[h \times w] \times 4}$ and $\boldsymbol{\delta}_i \in \mathbb{R}^{3 \times M}$ is a set of M 3D vectors, called offsets and defined in the equation 4:

$$\begin{aligned} \boldsymbol{\delta}_i &= \{\boldsymbol{\delta}_{1,i}, \boldsymbol{\delta}_{2,i}, ..., \boldsymbol{\delta}_{M,i}\} \\ &= \{\mathbf{Y}_1 - \mathbf{C}_i, \mathbf{Y}_2 - \mathbf{C}_i, ..., \mathbf{Y}_M - \mathbf{C}_i\} \end{aligned} \tag{4}$$

with $\mathbf{Y}_j \in \mathcal{S}_{\mathbf{T}_i} \quad \forall j \in [1, M]$, $\mathbf{T}_i$ the pose of the object visible in the ith patch and $\mathbf{C}_i \in \mathbb{R}^3$ is defined by :

$$\mathbf{C}_i = \begin{pmatrix} \frac{\mathbf{u_i} - c_x}{f_x} Z_i \\ \frac{\mathbf{v_i} - c_y}{f_y} Z_i \\ Z_i \end{pmatrix} \tag{5}$$

with f_x, f_y, c_x, c_y the camera intrinsics, $(\mathbf{u_i}, \mathbf{v_i})$ the 2D position of the center of the ith patch and Z_i the value of the patch depth at location $(\mathbf{u_i}, \mathbf{v_i})$. The equation 5 corresponds to the 3D backprojection of the 2D center of the ith patch, using a pinhole model. Thus, $\boldsymbol{\delta}_i$ is a set of M vectors, each one going from the 3D center of the patch and one of the M points in $\mathcal{S}_{\mathbf{T}_i}$. An example of offsets is visible figure 4: for 3 patches extracted in the image, we show $M = 9$ offsets. The use of offsets is very interesting for object pose estimation for two reasons. First, offsets bring invariance translation that is necessary due to the fact that we consider local patches. Indeed, 2 patches extracted from 2 different images with different poses may be very resembling. If displacement vectors are not used, the difference in terms of pose can thus only be seen as noise by the network. On the contrary, if offsets are employed the variable to regress is more correlated to patches aspect. Second, let's consider the space of all possibles object translation denoted Ω_t, if we do not use offsets then this space is at most $\mathbb{R}^3$. However when considering displacement vectors, the set of all possible offsets has an upper bound of D where D is the largest diameter of the object, thus $\Omega_t \subseteq \Omega_\delta = B(0, D)$ where $B(0, D)$ is the ball of center 0 and radius D and necessarily we have $\Omega_\delta \subset \mathbb{R}^3$. The space of possibilities being smaller when using offsets it is easier for a data driven algorithm to learn it.
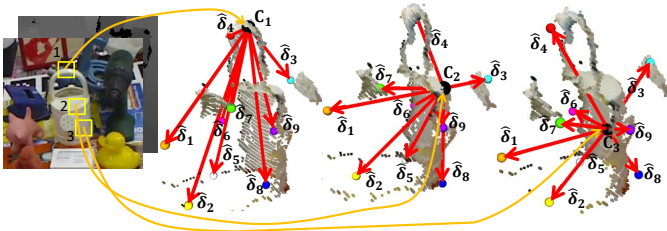

Fig. 4: Example of 3 patches extracted from an RGB-D image with the estimated offsets for each patch in red and the corresponding points represented by colored spheres, the center of each patch is shown by a black sphere.

Training. We denote the regression function g_{θ_r} where θ_r is the vector of weights of the network. The regression parameters are optimized by minimizing over the training data set:

$$\theta_r^* = \arg\min_{\theta_r} \mathcal{L}_r(\boldsymbol{\delta}, \hat{\boldsymbol{\delta}}) \tag{6}$$

where $\hat{\boldsymbol{\delta}} = g_{\theta_r}(\mathbf{I})$ and $\mathcal{L}_r$ is the mean absolute error:

$$\mathcal{L}_r(\boldsymbol{\delta}, \hat{\boldsymbol{\delta}}) = \frac{1}{3M} ||\boldsymbol{\delta} - \hat{\boldsymbol{\delta}}||_1 \tag{7}$$

where $||.||_1$ is the usual L^1 norm, thus $\mathcal{L}_r$ represents the averaged L^1 distance between the estimated and ground truth points. The L^1 norm is preferred to the L^2 norm because it is less sensitive to outliers, that are robustly handled during the voting step.

Inference and voting. Given the patches extracted in sec. 3.1 and their associated estimated probability, we discard the patches which probability is lower than a threshold τ. Thus we get a set: $\mathcal{P}_\tau = \left\{ \mathbf{I}_i | \hat{b}_i > \tau, i \in [1, K] \right\}$ that can be written: $\mathcal{P}_\tau = \{\mathbf{I}_i, i \in [1, N]\}$. For the ith patch $\mathbf{I}_i$ fed to the regression network we get M predicted 3D offsets $g_{\theta_r^*}(\mathbf{I}_i)$ denoted $\hat{\boldsymbol{\delta}}_i = \left\{ \hat{\boldsymbol{\delta}}_{j,i}, j \in [1, M] \right\}$. We can then get an estimation of the 3D location of the transformed points by adding the position $\mathbf{C}_i$ of the 3D center of the patch obtained from the equation 5. This way we get a set of M estimated points positions $\left\{ \hat{\mathbf{Y}}_{j,i}, j \in [1, M] \right\}$ in $\mathcal{F}_c$. When we take in account all the N patches we get $N \times M$ points: $\hat{V} := \left\{ \hat{\mathbf{Y}}_{j,i}, i \in [1, N], j \in [1, M] \right\}$ that can be viewed as M clusters of N points or votes in the $\mathcal{F}_c$. We denote the jth cluster of points $\hat{V}_j := \left\{ \hat{\mathbf{Y}}_{j,i}, i \in [1, N] \right\}$. The votes must then be aggregated to get a robust estimate of the 3D position of each point in the $\mathcal{F}_c$. We denote the aggregation function $h : \mathbb{R}^{3 \times N} \longmapsto \mathbb{R}^3$. It is necessary to aggregate the N 3D votes in a robust manner to limit the impact of possible outliers, hence h is chosen to be a non-parametric estimator of the maxima of density. In our case we use a mean-shift estimator [23,24] which iteratively estimates the local weighted mean in equation 8:

$$m(\mathbf{X}) = \frac{\sum_i k(\mathbf{X}_i - \mathbf{X})\mathbf{X}_i}{\sum_i k(\mathbf{X}_i - \mathbf{X})} \tag{8}$$

where k is a kernel function such as a Gaussian kernel: $k_\sigma(\mathbf{X}, \mathbf{Y}) = \exp(-\frac{||\mathbf{X} - \mathbf{Y}||^2}{2\sigma^2})$. Thus we can define the set $\hat{\mathcal{S}}_T := \left\{ \tilde{\mathbf{Y}}_j := h(\hat{V}_j), \forall j \in [1, M] \right\}$ which corresponds to the aggregated centroid of each cluster in $\mathcal{F}_c$. We show figure 5 such examples of votes.

3.3   3D-3D Correspondence alignment

In this section we show how to retrieve the pose using the estimated 3D keypoints in $\mathcal{F}_c$ that we obtained in the previous step and their corresponding reference keypoints in $\mathcal{F}_w$. Once the centroids have been voted we align the estimated points and their corresponding reference to get a pose estimation from the estimated location of the points. To do so we seek to find the transformation $\mathbf{T}^* \in SE(3)$ that minimizes:

$$\mathbf{T}^* = \arg\min_{\mathbf{T}} \sum_{j=1}^{M} ||\mathbf{T}\mathbf{X}_j - \tilde{\mathbf{Y}}_j||_2^2 \tag{9}$$

where $\mathbf{T}$ can also be represented with a minimal representation $q \in se(3)$, $X_j \in \mathcal{S}$, $\tilde{Y}_j \in \hat{\mathcal{S}}_T$ and $||.||_2$ is the euclidean norm of $\mathbb{R}^3$. That is finding the pose that best fits the points estimated by the aggregation of votes in $\mathcal{F}_c$. This problem is called the Orthogonal Procrustes Problem and can be solved using
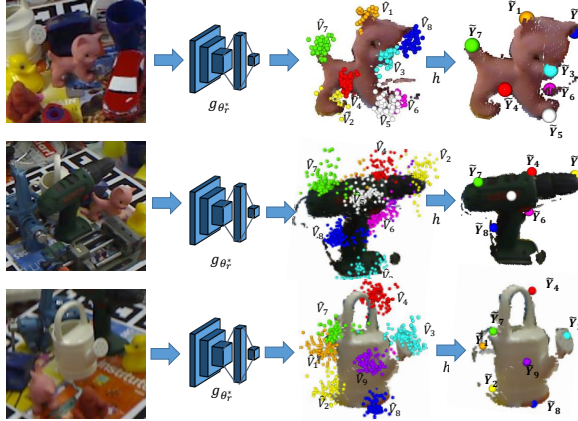
Fig. 5: 3 examples of predicted 3D points. From left to right: the cropped input image, the cluster of 3D votes $\hat{V}_j$ where each color corresponds to votes for a single point, the aggregated points $\tilde{\mathbf{Y}}_j$ obtained by mean-shift (best seen in color).

SVD decomposition as shown in [25] or an Iteratively Reweighted Least Square algorithm [26,27] to discard outliers and obtain a robust estimation. To further refine the pose we can apply an Iterative Closest Point (ICP) algorithm [28]. This consists in solving the equation 9, using the 3D model points and the points measured by the RGB-D camera projected in 3D using the equation 5.

4   Experiments.

We now present the results we obtained on the LineMod [4] dataset. This section is divided in four parts: first we present the technical details of our implementation, then we evaluate our method in terms of classification accuracy, 3D points regression accuracy and we measure the object pose accuracy using a standard metric and compare it to state-of-the-art results. Finally we compute the average inference time for a given object and study the impact of some hyper-parameters such as the level of patch density on both pose accuracy and inference time.

4.1   Implementation details

Training. To build our training data we extract patches in a sliding window fashion. To avoid a strong class imbalance between object and background patches and at the same time to get as much scene variance as possible we chose to limit the negative versus positive patches ratio to 9. We train the classification network for 100 epochs and the regression network for 500 epochs. We use a learning rate of 10^{-4} with the Adam optimizer. To avoid overfitting we stop the training process if the loss does not improve for 30 epochs and we recover the best weights found. We use data augmentation on the RGB data, randomly modifying the image brightness. A dropout of 50% is used for the classification CNN and and dropout of 20% and 10% is used on the two first fully connected layers

of the regression CNN. We implement the neural networks using the tensorflow-keras [29,30] framework.

Keypoints selection. Inspired by [7], we select the keypoints using the farthest point sampling algorithm which allows us to get a good coverage of the object. In our experiments we chose to use 9 points.

Inference. Our algorithm is implemented in Python. During the inference we extract patches with a stride of 4. We chose to set the threshold τ at 0.98 to avoid getting too many false positive that could pollute the voting space. The votes are aggregated using the mean-shift algorithm and a gaussian kernel with variance $\sigma^2 = 40^2 mm$. To recover the 6 DoF pose we use numpy SVD [31]. We use as well open3d ICP [32], on the sub map defined by the estimated bounding box. For testing we use a Nvidia RTX2070 and an Intel Xeon @3.7 GHz.

4.2   Datasets

The LineMod dataset consists of about 15 000 RGBD images of 13 objects with multiple recorded poses in a variable and cluttered environment. It is widely used by the pose estimation community. We use the same method as [2,6,7] to select training and testing images. These constraints lead to small datasets, however with the patch based approach and using data augmentation on the luminosity we do not need to train on synthetic data. In our experiments we do not use symmetric objects like glue and eggbox and focus on the 11 remaining objects.

4.3   Classification accuracy

In this subsection, we measure the performance of the classification network. Having a good accuracy is necessary before trying to regress the position. Indeed having a bad classification accuracy could lead to multiple patches being misclassified. A high false positive rate would create noise in the Hough space and complexify the task of finding the maximum density. On the contrary, a high false negative rate would reduce the number of patches used for regression and thus the number of votes, leading to a less robust estimation. As many more patches are extracted from the background than the foreground, it is preferable to have a higher true negative rate than true positive rate. It is important to note that the size of patches is an important hyper parameter that needs to be chosen carefully. With our strategy the number of patches extracted from the image can be large if the sliding window step is small, hence requiring many inferences to be done by the CNN which is computationally intensive. To speed up the detection we first extract patches from the image in a non overlapping way and feed them to the convolutional network. This allows us to get a rough estimate of the object bounding box. We can then extract again patches but in a dense way from the area defined by the bounding box. We focus here on the first classification step used to estimate the 2D bounding box, using a stride of 8 and a threshold of 90%. We can see that for every object we get a high true negative rate above 99.6 % meaning we do not pollute the vote space. The true positive rate is much more variable, from 87.2 % to 96.9 % but stays in the

high range, so not too many patches are discarded. A noticeable exception is the lamp that gets an accuracy of 66.7% which can be explained by its lack of local discriminating features.

	ape	ben.	cam	can	cat	drill.	duck	hole.	iron	lamp	phone	MEAN
True pos.	96.9	92.3	85.0	92.0	96.2	94.0	92.5	93.5	94.2	66.7	87.2	90.0
True neg.	99.8	99.7	99.6	99.8	99.8	99.8	99.9	99.8	99.7	99.8	99.7	99.8

Table 1: True positive and true negative rate (in %) for each object using our classification network.

4.4 3D Points regression accuracy

In this subsection, we study the accuracy of the regression network. For each object we measure the average euclidean distance between the estimated position of each keypoint after it has been aggregated and its ground truth position.

	ape	ben.	cam	can	cat	drill.	duck	hole.	iron	lamp	phone	MEAN
Average error (mm)	10.7	11.2	27.9	12.1	10.1	12.6	9.1	12.0	15.2	18.2	12.5	13.8
Median error (mm)	10.7	10.8	27.7	12.8	10.1	11.5	9.2	12.6	14.0	17.6	12.9	13.6
Min. avg. err. (mm)	8.9	7.2	24.1	9.2	6.3	8.2	6.7	8.9	11.5	12.8	9.1	10.3
Max. avg. err. (mm)	13.2	16.7	31.7	14.1	13.9	20.0	11.7	14.8	21.4	25.2	16.6	18.1

Table 2: Average, median, minimum and maximum euclidean distance (in mm) for each object.

We can see that the euclidean distance between predicted and ground truth points is comprised between 6.7 mm and 31.7 mm with an average of 13.8 mm. While most objects get an average error of 10 to 12 mm and seemingly proportional to their diameter, we obtain large errors with the cam that can be an explanation of the coarse results we obtain in table 3.

4.5 Object pose accuracy

Metric. We use the standard 6 DoF metric developed in [4], the average distance of model points (ADD). A pose is considered correct if the value of the ADD is less than 10% of the object diameter D. We report the results in table 3. As we can see, our method improves state-of-the-art results on average by 1.6 % without applying an ICP refinement. When an ICP algorithm is used results are improved by 5.5 % on average. We can see that [7] obtains almost perfect results on some objects while failing on a few objects like ape, cat and duck, which are small objects. On the other hand [2] gets very good results on almost every object, barely failing on the duck and the cam. Our method however gets the best of both worlds, barely failing on some small challenging objects like ape and duck and getting above 95% accuracy on most objects. We can see that the results we obtain in terms of pose are linked to the points prediction and patch

Input	RGB		RGB-D			
Method	BB8 [6]	w. ref. PVNet [7]	DenseFusion [2]	w. ref.	Ours	Ours + ICP
ape	40.4	43.6	92.0		87.9	**93.7**
ben.	91.8	**99.9**	93.0		99.7	99.8
cam	55.7	86.9	**94.0**		89.5	98.0
can	64.1	95.5	86.0		94.8	**99.3**
cat	62.6	79.3	93.0		97.6	**99.2**
drill.	74.4	96.4	97.0		97.8	**99.5**
duck	44.3	52.6	87.0		85.0	**94.7**
hole.	67.2	81.9	92.0		90.5	**98.1**
iron	84.7	98.9	97.0		98.0	**99.6**
lamp	76.5	**99.3**	95.0		98.1	99.0
phone	54.0	92.4	93.0		97.2	**98.3**
MEAN	65.1	84.2	92.6		94.2	**98.1**

Table 3: Percentage of correctly predicted poses using the ADD metric on the LineMod dataset with state-of-the-art methods.

classification: objects that get a large points error relatively to their diameter (e.g. cam) tend to fail with the ADD metric.
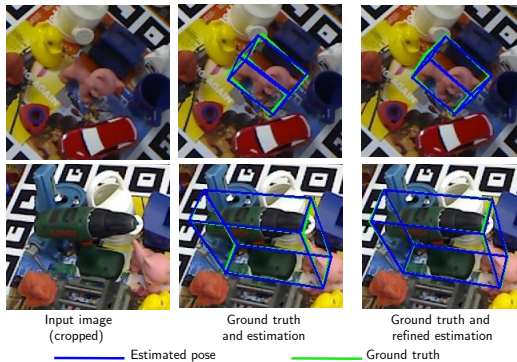

Fig. 6: Some qualitative examples of pose estimation results on LineMod. The pose estimation is represented as the blue bounding box and the ground truth pose as the green bounding box.

4.6   Inference time

Inference time is greatly dependent on the choice of the density with which patches are extracted. The lower the stride is, the more patches have to be extracted and fed to the networks and the longer the inference will be. However we expect the accuracy to be growing with the number of patches extracted. This balance allows our method to be suitable to a wide range of methods. The flexibility it brings lets the user tune the extraction stride to better meet the application needs. In table 4 we analyze inference time for the driller and in figure 7 we study the balance between inference time and accuracy, using different strides for patch extraction. We can tune both the stride of patch extracted

to estimate the bounding box and the stride of patches extracted within the bounding box. To report the results in table 3 we have used low strides which explains why the times reported in the first line of table 4 are quite high, however as we can see in figure 7 we can use greater strides to diminish testing times while still having a very good accuracy. We do not report in this section inference times and accuracy using the ICP refinement.

	1st extrac.	Bbox	2nd extrac.	Class.	Reg.	Voting	3D-3D solving	Total
Time (ms) (8/4)	228.3	640.4	311.1	724.8	180.8	758.2	0.4	2844
Time (ms) (48/16)	6.9	24.2	6.5	22.5	6.1	46.1	0.5	112.8

Table 4: Inference times (in ms) for the driller, for each step of our pipeline, for different strides m/n where m is the first stride extraction used to estimate the bounding box and n is the second one (in pixels).

As we can see in table 4, one bottle neck of our code is the mean-shift aggregation. However it is important to note that we could easily process the mean-shifts in parallel as we do 9 different mean-shifts on independent clusters of points.
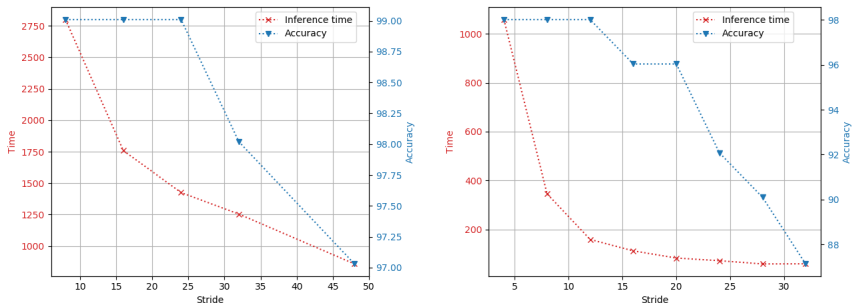

Fig. 7: Inference time (in ms) and accuracy for the driller for varying strides (in pixels) for the bounding box estimation (left, with second stride of 4) and the second classification (right, with first stride of 48).

5   Conclusion

In this paper, we introduced a novel approach to estimate 6-DoF object pose in a RGB-D image. Our method leverages the strengths of patch voting based strategies and hybrid learning-geometrical methods, using patches extracted from the image to predict a set of sparse 3D keypoints representing the object geometry in $\mathcal{F}_c$. Those points are then put in correspondence and aligned with reference keypoints to retrieve the pose. We showed that our strategy is more robust and accurate than state-of-the-art.

References

1. Marchand, E., Uchiyama, H., Spindler, F.: Pose estimation for augmented reality: a hands-on survey. IEEE transactions on visualization and computer graphics **22**(12) (2015) 2633–2651
2. Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., Savarese, S.: Densefusion: 6D object pose estimation by iterative dense fusion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 3343–3352
3. Tejani, A., Tang, D., Kouskouridas, R., Kim, T.K.: Latent-class Hough forests for 3D object detection and pose estimation. In: European Conference on Computer Vision, Springer (2014) 462–477
4. 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, Springer (2012) 548–562
5. Kehl, W., Milletari, F., Tombari, F., Ilic, S., Navab, N.: Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. In: European Conference on Computer Vision, Springer (2016) 205–220
6. Rad, M., Lepetit, V.: BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In: Proceedings of the IEEE International Conference on Computer Vision. (2017) 3828–3836
7. Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: PVNet: Pixel-wise Voting Network for 6DoF pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 4561–4570
8. Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6D object pose prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 292–301
9. 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 International Conference on Computer Vision. (2019) 7678–7687
10. Park, K., Patten, T., Vincze, M.: Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 7668–7677
11. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: DeepIM: Deep Iterative Matching for 6D pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 683–698
12. 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)
13. Tekin, B., Bogo, F., Pollefeys, M.: H+O: Unified egocentric recognition of 3D hand-object poses and interactions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 4511–4520
14. Riegler, G., Ferstl, D., Rüther, M., Bischof, H.: Hough networks for head pose estimation and facial feature localization. Journal of Computer Vision **101**(3) (2013) 437–458
15. Fanelli, G., Gall, J., Van Gool, L.: Real time head pose estimation with random regression forests. In: CVPR 2011, IEEE (2011) 617–624
16. Kacete, A., Royan, J., Seguier, R., Collobert, M., Soladie, C.: Real-time eye pupil localization using Hough regression forest. In: 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE (2016) 1–8

17. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. IEEE transactions on pattern analysis and machine intelligence **33**(11) (2011) 2188–2202

18. Do, T.T., Cai, M., Pham, T., Reid, I.: Deep-6DPose: Recovering 6D object pose from a single RGB image. arXiv preprint arXiv:1802.10367 (2018)

19. 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. (2017) 1521–1529

20. Sundermeyer, M., Marton, Z.C., Durner, M., Brucker, M., Triebel, R.: Implicit 3D orientation learning for 6D object detection from RGB images. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 699–715

21. Mahendran, S., Ali, H., Vidal, R.: 3D pose regression using convolutional neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. (2017) 2174–2182

22. Pavlakos, G., Zhou, X., Chan, A., Derpanis, K.G., Daniilidis, K.: 6-DoF object pose from semantic keypoints. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2017) 2011–2018

23. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Transactions on pattern analysis and machine intelligence **24**(5) (2002) 603–619

24. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE transactions on pattern analysis and machine intelligence **17**(8) (1995) 790–799

25. Arun, K.S., Huang, T.S., Blostein, S.D.: Least-squares fitting of two 3-D point sets. IEEE Transactions on pattern analysis and machine intelligence (5) (1987) 698–700

26. Fitzgibbon, A.W.: Robust registration of 2D and 3D point sets. Image and vision computing **21**(13-14) (2003) 1145–1153

27. Malis, E., Marchand, E.: Experiments with robust estimation techniques in real-time robot vision. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE (2006) 223–228

28. Besl, P.J., McKay, N.D.: Method for registration of 3-D shapes. In: Sensor fusion IV: control paradigms and data structures. Volume 1611., International Society for Optics and Photonics (1992) 586–606

29. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015) Software available from tensorflow.org.

30. Chollet, F., et al.: Keras. `https://keras.io` (2015)

31. Oliphant, T.: NumPy: A guide to NumPy. USA: Trelgol Publishing (2006–)

32. Zhou, Q.Y., Park, J., Koltun, V.: Open3D: A modern library for 3D data processing. arXiv:1801.09847 (2018)