Evaluation of the Accuracy of Pose Estimation Based on Relative Pose

Francisco Lourenço
francisco.rlourenco@gmail.com
Helder Araújo
helder@isr.uc.pt

Institute of Systems and Robotics
Department of Electrical and Computer Engineering
University of Coimbra

Abstract

In this paper we describe an approach for the evaluation of the estimation of absolute pose. The proposed approach is based on the estimation of the relative pose and on the computation of a metric based on the relative rotation and relative translation between objects. When multiple objects are present in a scene and if only the camera moves, their relative poses remain constant between frames. Using the absolute poses of the objects, the relative poses in each frame can be estimated and their variation can be used to evaluate the algorithms. One of the advantages of use of the relative pose is that it can be applied even if a ground truth pose is not available, e.g., in pose estimation approaches without object recognition.

1 Introduction

Absolute pose of both RGB and RGB-D images has been addressed using machine learning approaches. A comprehensive review of 6DoF object pose estimation can be found in [1]. 6DoF pose estimation can be performed at two different levels:

- Instance-level - When a method is said to work at the instance-level, it means that, to estimate the 6DoF pose of an object, such a method will estimate the pose of a known object instance, i.e., an object used in the training phase.

- Category-level - As opposed to instance-level, category-level approaches deal with unseen objects. Instead of precisely identifying the object they want to look for, these methods work with categories, e.g., cars, bicycles, boxes, toys. For example, while at the instance-level, the methods know precisely the brand, model, and color of a car whose pose they want to estimate, at category-level, the only information the methods have is that they should look for a car. Estimating 6DoF object pose at the category-level is usually a more complex problem to solve, but methods that work at this level generalize better than at the instance-level, as it might be possible to estimate the pose of a broader range of unseen objects instances.

Furhtermore 6DoF pose estimators can be divided into two main categories:

- 3D bounding box detectors - These methods work at the category-level and do not estimate the 6DoF pose directly but, instead, they fit a 3D bounding box to the object. To do so, these methods produce oriented 3D BBs (Bounding Boxes) centered at some point $x = (x, y, z)$, size $d = (d_w, d_h, d_l)$ with orientation (θ_y). The bounding box can then be extended to the 6DoF space where the pose can be recovered.

- Full 6DoF pose estimators - Directly estimate the 3D translation and 3D rotation. Typically, these methods work at the instance-level. However, recent proposed full 6DoF pose estimators such as [2] address the category-level problem.

Several approaches for pose estimation exist. [3] is an instance-level/ classification/ full pose estimator. This method was originally designed to estimate poses from RGB images but the authors also present in their work a variant where they adapt it by opening a depth channel and consequently estimating the 6DoF pose from RGB-D inputs. [4] is an instance-level/template matching/full pose estimator. This method uses a SVM and templates that are used for object detection and, if a 6DoF pose is assigned to the templates, these can vote for the 6DoF pose of an object instance. [5] is an instance-level/template matching/full pose estimator. This approach is quite simple and focuses on solving the problem of template matching-based methods where the templates are built online from the RGB-D outputs and require physical interaction of a human operator or a robot with the environment.

In the work described in this paper we decided to evaluate the approach DenseFusion [6], which is an instance-level/ regression/ full pose estimator. The core idea of this approach is to embed and fuse RGB values and point clouds at the per-pixel level. The goal is to fully leverage RGB and depth information. The use of the depth information as an extra channel of the RGB image is not considered since these data are defined in different domains. In DenseFusion a heterogeneous architecture is proposed, where RGB data and depth information are processed individually and then densely fused at the per-pixel level, where each extracted feature will vote for a 6DoF pose.

2 Methodology

This study aims to infer the 6DoF pose estimator accuracy in terms of the relative pose of the objects present in the image. Having multiple images of the same scene but acquired from different positions of the camera, the absolute poses of the objects change. However, their relative poses with respect to each other remain constant. Hence, by measuring how these relative poses change between frames, it is possible to infer the quality of a pose estimation model, not only by measuring how these relative poses change between frames, but also from the ground truth poses given on a dataset.

2.1 Proposed Metric for 6DoF Pose Estimator Precision Measurements

Having a set of frames of size N, each containing k objects, with $N > 1$ and $k > 1$, and the respective poses, and if we consider that the camera acquires images of all objects in every frame, an object can be randomly chosen as a reference, which will be denoted as object o_r. Then, the poses of the other objects relative to the reference object are calculated. This is done for all frames and will result in $k-1$ relative transformations per frame. These transformations are equal in all frames under the assumption that the objects did not move during image acquisition. Therefore, the differences between corresponding relative poses in different frames can be estimated to characterize the accuracy of the approach.

If we write a 3D transformation as:

$$[R|t] = \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

We can compute the relative transformation between the reference object o_r and object j in frame n as follows:

$$n[R|t]_{o_r j} = n[R|t]_{o_r}^{-1} \cdot n[R|t]_j \tag{2}$$

This is done for all frames and, when all relative 3D transformation matrices $n[R|t]_{o_r j}$ are available, the error between these matrices is measured between relative transformations in the frames by calculating the product of the inverse of matrix $(n-1)[R|t]_{o_r j}$ by matrix $n[R|t]_{o_r j}$. Essentially, we are calculating the relative transformation between relative transformations and, if these are the same, the result should be an identity matrix of size $4*4$. Hence, the error will be the Frobenius norm of the difference between the resulting matrix and an identity matrix, as expressed in the following equation:

$$n\varepsilon_{[R|t]_{or\,j}} = ||I_4 - (n-1)[R|t]_{or\,j}^{-1} \cdot n[R|t]_{or\,j}||_f \quad (3)$$

Where $n\varepsilon_{[R|t]_{or\,j}}$ is the j^{th} element of error vector $n\varepsilon_{[R|t]}$ (vector containing all $k-1$ transformation errors of consequent frames) between frames n and $n-1$. This will yield N-1 error vectors of size $k-1$ that, in the end, are averaged into a single precision score. This score will be denoted as $\tau_{[R|t]}$ and can be seen as how much the poses "jitter" between frames. The smaller it is, the better. The use of the relative coordinate transformation instead of the Frobenius norm of the difference between matrices is due to the fact that this measure corresponds to the pose "difference" measured on the manifold. The object to be used as a reference can be any of the objects from the training set. It is selected "a priori" allowing for the automatic computation of the metric.

2.2 The Proposed Metric as a Loss Function

This metric can also be used as a loss function. In fact the similarity between relative poses can also be used for training a machine learning model. Therefore, if we consider equation 5, and substitute the relative poses by an estimated and a target pose, we can infer their similarity. This function can then be back-propagated and hence used to update the weights of a neural network.

For that purpose the proposed metric can now be rewritten as:

$$L = ||I_4 - [R|t]_{target}^{-1} \cdot [R|t]_{estimated}||_f \quad (4)$$

For DenseFusion, the equation can be written as follows, where a confidence score is also added so that the model can learn in a self-supervised fashion:

$$L = \frac{1}{N_{Features}} \sum_i (c_i \times ||I_4 - i[R|t]_{target}^{-1} \cdot i[R|t]_{estimated}||_f - \omega \times log(c_i)) \quad (5)$$

When compared with the ADD(-S) based loss function, a significant advantage of this metric is that it does not require the objects 3D models. For that reason, the proposed loss function can be less computationally intensive, thus improving training time.

2.3 Results

In this work the YCB-Video dataset was used to evaluate the accuracy. This dataset was proposed in[7] and it provides accurate 6D poses of 21 objects observed in 92 videos with 133,827 frames.The poses estimated by DenseFusion and the ground truth poses of the YCB-Video dataset were used to evaluate the accuracy. However, this metric can be used for any dataset or 6DoF pose estimator, as long as the images acquired correspond to the same scene with multiple object instances that did not move while the images are obtained.

===	YCB-Video	DenseFusion Original
$\tau_{[R\|t]}$	0.0011	0.0887
====		

Table 1: Relative pose study results (accuracy).

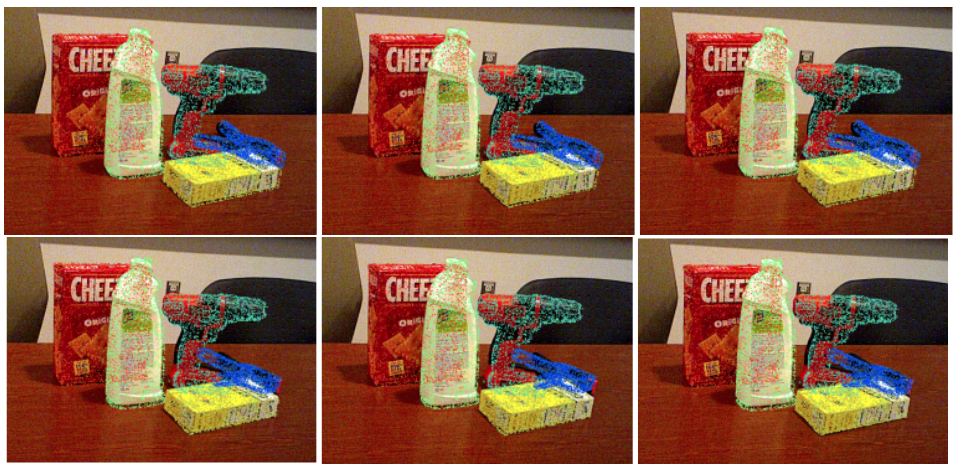

Figure 1: YCB ground-truth (1st row); DenseFusion (2nd row) poses.

The results shown here were obtained for the evaluation videos from the YCB-Video dataset, totaling 20738 frames.

As expected, the YCB-Video Dataset labels are the most precise poses. These results can be seen as how much jitter there is on the estimated poses between consequent frames. Visually, if we fit the 3D models to the RGB images, a low score will result in a smoother representation, while with high scores, the models look like they vibrate during the video. So, this score can be seen as how much ”vibration” it is on the projected 3D models. Figure 1 illustrates this.

References

[1] Caner Sahin, Guillermo Garcia-Hernando, Juil Sock, and Tae-Kyun Kim. A Review on Object Pose Recovery: from 3D Bounding Box Detectors to Full 6D Pose Estimators. jan 2020.

[2] Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-PACK: Category-level 6D Pose Tracker with Anchor-Based Keypoints. oct 2019.

[3] Eric Brachmann, Frank Michel, Alexander Krull, Michael Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image. In *CVPR*, volume 2016-Decem, pages 3364–3372. IEEE, jun 2016.

[4] Reyes Rios-Cabrera and Tinne Tuytelaars. Discriminatively Trained Templates for 3D Object Detection: A Real Time Scalable Approach. In *ICCV*, pages 2048–2055. IEEE, dec 2013.

[5] S. Hinterstoisser and V. Lepetit and S. Ilic and S. Holzer and G. Bradski, K. Konolige, and N. Navab. *Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes*, volume 7584 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[6] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martin-Martin, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion. In *CVPR*, volume 2019-June, pages 3338–3347. IEEE, jun 2019.

[7] You Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. Nov 2017.