# Using PCL Gobal Descriptors in a DenseFusion Architecture

Nuno Pereira
nuno.pereira@ubi.pt
Luís A. Alexandre
luis.alexandre@ubi.pt

Departamento de Informática
Universidade da Beira Interior
Instituto de Telecomunicações
6201-001 Covilhã, Portugal

## Abstract

In this paper, we present an alternative architecture to the state-of-the-art in 6D pose - DenseFusion. We changed the architecture of the method in the depth feature extraction phase. Instead of using the PointNet, as used in the original DenseFusion, we used global descriptors from the Point Cloud Library (PCL) to extract features. We made a comparison in terms of average accuracy between the Ensemble of Shape Functions (ESF), Viewpoint Feature Histogram (VHF) and the original PointNet.

## 1 Introduction

Object detection and pose estimation is an important problem in the computer vision domain, for which many solutions have been proposed. Object detection and pose estimation of objects is a fundamental task due to its use in many different areas, ranging from robotics to augmented reality and many others.

6D pose of an object is the representation of its position in space (x, y, z) and orientation in each one of the axis. This type of object pose is usually represented by a Rotation matrix, $R$, and a Translation vector, $t$. With this data, we can know the placement and orientation of an object in the scene or we can place virtual objects in scenes and both these motivations justify the importance of this area in robot grasping and augmented reality.

In this paper we use DenseFusion [5] which is one of the state-of-the-art solutions to achieve 6D pose estimation of an object. This method performs a fusing step of 3D data with 2D appearance features while retaining the geometric structure of the input space. The authors present results where this method outperforms PoseCNN [7] on the YCB-Video dataset without the post-processing. DenseFusion in its base is more similar to PointFusion [8], in which geometric and appearance information is fused in a heterogeneous architecture. DenseFusion is a generic framework for 6D pose estimation of a set of known objects from RGB-D images. It has a heterogeneous architecture that processes two data sources individually and uses them to extract pixel-wise features, from where the pose is estimated. The method also contains an iterative pose refinement procedure that further improves the pose estimation.

DenseFusion architecture can be divided into four phases as shown in Figure 1. The first one is the step where the method receives the raw RGB data and applies object segmentation to get the masks that represent the objects in the scene. The second phase is the feature extraction, where features are extracted from the RGB and depth images. After the feature extraction, both the features of the RGB images and the depth are fused in a pixel-wise manner in the third phase, and immediately after, the pose predictor estimates the pose of each object in the scene, giving as output the rotation matrix and the translation vector. Finally, on the last phase, the pose refinement executes small adjustments on the poses of the objects and returns the final results. Denote that this last phase - the refinement phase, is optional.

One of the most important phases of DenseFusion architecture is the feature extraction phase, where features are extracted from RGB images and depth. In the original architecture of DenseFusion, the authors use a fully convolutional neural network (FCNN) to extract features from the RGB images and use the PointNet [3] method to extract features from the depth. These extraction procedures and their fusion is shown in Figure 2.

This work was supported by project 026653 (POCI-01-0247-FEDER-026653) INDTECH 4.0 – New technologies for smart manufacturing, cofinanced by the Portugal 2020 Program (PT 2020), Compete 2020 Program and the European Union through the European Regional Development Fund (ERDF).

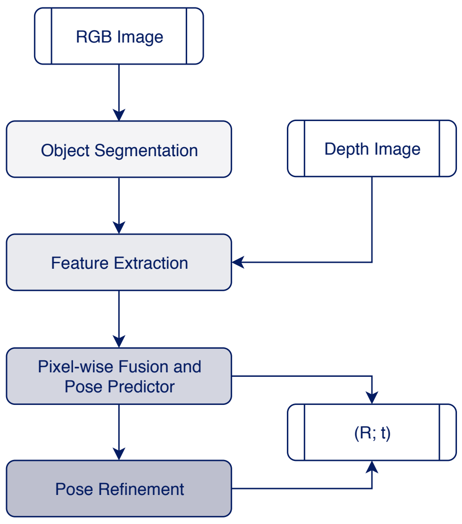

Figure 1: Overview of DenseFusion architecture.

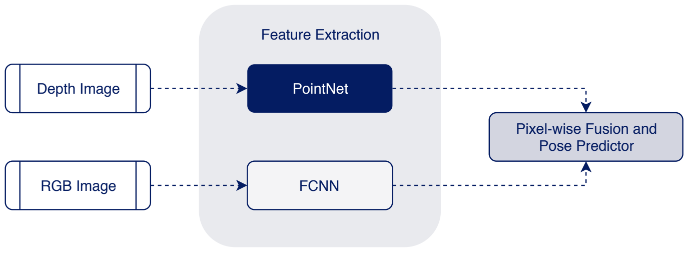

Figure 2: DenseFusion feature extraction phase.

## 2 Proposed Method

What we propose in this paper, is a modification in the architecture of the DenseFusion, wherein the feature extraction phase instead of using the PointNet method, uses PCL global descriptors to extract features from the depth of the objects and then fuses these features with the RGB features. The main reason to choose global descriptors is the fixed size of the features vector. We tested two global descriptors available in PCL, Ensemble of Shape Functions [6] (ESF) and Viewpoint Feature Histogram [4] (VFH). We changed the blue block of Figure 2. This change required us to change the first layer of the neural network where the pixel-wise fusion occurs so it could receive the output from the ESF and VFH because the size of this output differs from the original size that the DenseFusion CNN was capable of receiving. This change was require due to the fact that FCNNs require a fixed-size input length.

Ensemble of Shape Functions (ESF) is a global descriptor that consists of 10 concatenated 64-bin histograms. It outputs a single normalized histogram with a size of 640 for a point cloud. We choose the ESF descriptor from the PCL to extract the features of the point cloud generated from the depth data that we also receive from the RGB-D cameras. Our main reason to choose this descriptor is that it does not need the calculations of the normals and this helps our method in terms of the performance and it is a global descriptor [1].

The second global descriptor that we have tested is the Viewpoint Feature Histogram (VFH) that differentiates itself from other descriptors by using the viewpoint vector direction. It consists of four 45-bin histograms for the angles and distances between each point and one 128-bin histogram that represents each points Normals. So the output of the descriptor has a single normalized histogram with 308 values for a point cloud. We choose to test the VFH descriptor from the PCL to extract the

| Method \ Epoch | Average Error (Standard Deviation) [mm] 10 | 20 | 30 | 40 | 50 | Avg. time to train (Stand. Deviation) [h] |
|---|---|---|---|---|---|---|
| DenseFusion using PointNet [3] | 12.9 (0.5) | 12.6 (0.5) | 12.4 (0.4) | 12.5 (0.4) | 12.8 (0.3) | 42 (0.4) |
| DenseFusion using ESF [6] | 13.4 (0.4) | 13.0 (0.2) | 12.9 (0.1) | 13.1 (0.4) | 13.2 (0.4) | 34 (1.1) |
| DenseFusion using VFH [4] | 13.1 (0.3) | 13.0 (0.3) | 12.8 (0.4) | 12.7 (0.3) | 12.9 (0.3) | 34 (1.1) |

Table 1: Results of our experiments. Average error and standard deviations in millimeters and average time to train in hours.

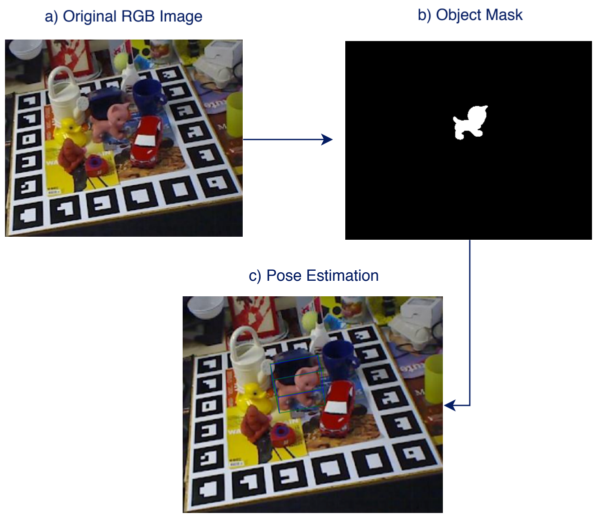

Figure 3: A short representation of the possible visible steps of the DenseFusion method using an image from the LineMOD dataset. Image *a* is an input example of a RGB image. Image *b* is the mask extracted after the object segmentation phase of a specific object. Image *c* is the input image with a box representing the pose prediction of the cat object.

features of the point cloud generated from the depth data because it is also a global descriptor and has fewer values than the ESF this could help on the size of the predictor neural network that we use after this phase [1].

## 3 Experiments

In our experiments, we used the LineMOD [2] dataset as input data to all of our experiments and this dataset is also used on most of the related work in 6D pose estimation. It contains 15 household objects. Each object is associated with a test image showing one annotated object instance with significant clutter but only mild occlusion. In Figure 3 we can see a visual representation of an example image from the LineMOD passing through the DenseFusion pipeline phases. All of our experiments were executed on a NVIDIA GEFORCE GTX 1080 Ti.

We trained each of the three different methods (PointNet, ESF and VFH) 50 epochs with tests occurring in epoch 10, 20, 30, 40 and 50. All the experiments were executed without the pose refinement phase of the DenseFusion architecture and we used the same standard hyperparameters as in [5] (bash size one, 0.0001 as learning rate value and Adam optimizer). The results obtained are presented in Table 1 and DenseFusion using PointNet as feature extractor from depth has the best error around $12mm$. PointNet is a method based on deep learning which provides a unified architecture for applications ranging from object classification, part segmentation, to scene semantic parsing, since it is a neural network it needs to be trained. When we use global descriptors like ESF and VFH they just process the data and output the unique features of the objects so we need less time to train all the pipeline of the DenseFusion. Using ESF and VFH methods instead of PointNet we have errors around $13mm$, but we needed 8 hours less to train the DenseFusion pipeline which represents a 19% faster approach. In terms of the inference time we achieved $14ms$ per image using ESF or VFH, comparing it with $15ms$ using PointNet.

In Figure 4 we present the error values in millimeters along the number of epochs of training. We can conclude that after 30 epochs the DenseFusion network starts over-fitting and starts to perform worst on the test part of the dataset. On the experience where we used VFH as depth feature extraction of the object, the network only started to over-fit after epoch 40.

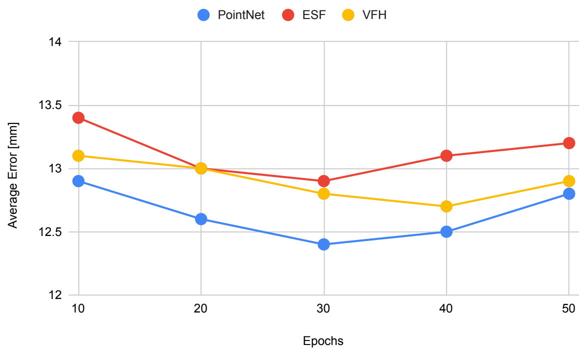

Figure 4: Average error in millimeters tested in different epochs.

## 4 Conclusion

Our experiments showed that 30 epochs are enough to train the DenseFusion. With these 30 epochs it is possible to achieve good results in terms of pose estimation and using as less time as possible during the training phase of the method. Pose refinement should be used after the 30 epochs of training to obtain a minor decrease in the position errors. Using global descriptors instead of PointNet can be an advantage due to the 19% less time needed to train DenseFusion with the cost of less than $1mm$ of error.

## References

[1] Luís A. Alexandre. 3D descriptors for object and category recognition: a comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October 2012.

[2] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. pages 858–865, November 2011.

[3] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[4] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162, October 2010. doi: 10.1109/IROS.2010.5651280.

[5] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3d object classification. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 2987–2992, Dec 2011. doi: 10.1109/ROBIO.2011.6181760.

[7] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *CoRR*.

[8] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018.