POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

6 DoF object pose estimation from vision for robotic grasping

Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Author: Claudia Speranza

Advisor: Prof. Matteo Matteucci

Co-advisor: Dr. Simone Mentasti

Academic year: 2021-2022

1. Introduction

6DoF pose estimation involves detecting the pose of an object in three-dimensional space, which includes the location and orientation with respect to a reference point of view. This is of crucial importance in the robotic field, where robots need to interact with objects in the environment. Robotic grasping is one of the fundamental and most challenging skills of robots, it requires the coordination of robotic perception, planning and control of the movements. Our work inserts itself into the first step of the process providing a system to determine the pose of objects in the surroundings. This datum is then fed to a solutor that computes the best trajectory to reach the object. At this point, the end-effector placed at the end of the robotic arm has all the information it needs to pick up the object. We propose a vision-based approach that involves the use of particular cameras, called *depth cameras*, for the task of detecting the object's pose. In addition to the well-known colored picture, depth cameras also provide distance information for each pixel in the color frame. Our solution involves the design of a neural network model that extracts the 6 DoF pose of an object, exploiting only the data provided from the depth camera.

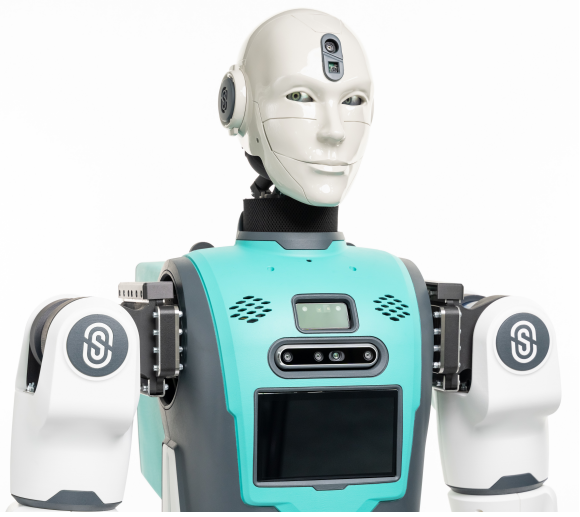

Figure 1: Upper body of a humanoid robot called "Robee", developed by Oversonic Robotics

This work has been realized as an application for *Oversonic Robotics*. Oversonic Robotics is an Italian robotics startup founded in 2021. Their first project is a humanoid robot called "Robee", shown in Figure 1. He operationally replicates the mechanical structure of the human body, he has arms and hands that allow him to cover all kinds of tasks, from the simplest gestures to a solid grip for handling objects. Robee is

equipped with a set of different cameras used for object detection to improve his awareness of the surrounding environment.
The main contributions of this work are as follows:

- A quantitative analysis and comparison of different depth cameras available on the market.
- A flexible deep network for 6 DoF object pose estimation that is capable of predicting translation and rotation independently exploiting images captured from a depth camera and that outperforms the state-of-the-art models.
- A novel definition of a metric that describes the accuracy of the predicted pose in case of symmetric objects.
- An empirical study of performances of models trained with synthetically generated data.

2. State of the art

In the computer vision sector, computing a 6DoF pose of an object is a challenging problem that involves detecting both three-dimensional orientation and position of an object with respect to some reference camera coordinates. Ideally, a solution should deal with objects of varying shape and texture, show robustness towards heavy occlusion, sensor noise, and changing lighting conditions, while achieving the speed requirement of real-time tasks. More recently, with the explosive growth of machine learning and deep learning techniques, Deep Neural Network based methods have been introduced into this task and reveal promising improvements. 6 DoF object pose estimation methods in the literature can be classified based on the input they receive. Traditionally colored images are used alone to estimate objects' pose, for instance PoseCNN network [5]. However, with the development of cheaper and more accurate RGB-D sensors, the depth information has been included in the models and higher performance has been reached. Researchers had to face the problem of combining heterogeneous data that come from color image and 3D point cloud. Xi et al. [6] proposed the PointFusion network that extracts point cloud features using a variant of PointNet and derives the image appearance features from a CNN. The two vectors of features are then combined in a fusion network to extract 3D bounding boxes. Later, Wang et al., [4], refer to the PointFusion network, introducing a novel local feature fusion scheme and a fast iterative refinement to further improve the pose estimation, generating a network called DenseFusion.

Figure 2: Intel RealSense D455 depth camera

3. Experimental setup

Standard digital cameras output images as a 2D grid of pixels. Each pixel has three values associated with it that define red, green and blue color components. Depth cameras, on the other hand, have pixels with a different numerical value associated with them, that number being the distance of the corresponding pixel from the camera, or "depth". Some depth cameras have both an *RGB* and a depth system, which can give pixels with four values, therefore are referred to as *RGB-D* images. There are a variety of different methods for calculating depth, all with different strengths and weaknesses and optimal operating conditions. We focus on stereo depth cameras for their use in our application. A stereo depth camera is a type of camera with two separate image sensors, the distance between them is called baseline. These sensors allow the camera to simulate human binocular vision and achieve the ability to capture depth. The distance these cameras can measure is directly related to how far apart the two sensors are: the wider the baseline is, the further the camera can see.
For this work, we employ the *Intel RealSense D455* depth camera. This specific model has a baseline of 95 mm, and its ideal range of action lies between 0.6m and 6m. The choice of this sensor has been performed after an accurate analysis of its accuracy in depth reconstruction. In our operating scenario, we positioned the sensor on the robot's chest. In this way, the camera is fixed to the robot structure, and we can compute the trajectory to reach the object based on images captured by it.

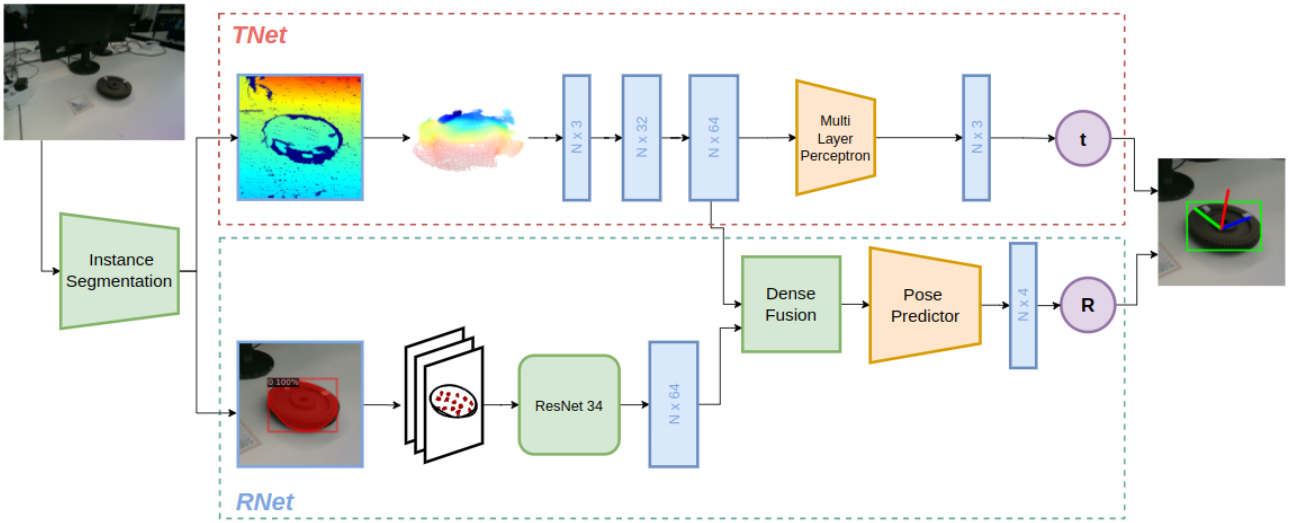

Figure 3: Model architecture

4. Model architecture

The objective of the network we propose is to estimate the 6 degrees of freedom pose of an object in the three-dimensional space with respect to the camera. The 6dof pose is commonly represented with a homogeneous transformation matrix, composed by a rotation matrix $R \in SO(3)$, being $SO(3)$ the group of all rotations around the origin of a three-dimensional Euclidian space, and a translation $t \in \mathbb{R}^3_x$. For computational reasons, we prefer to work with quaternions since they grant a more compact representation of the rotation in the form of a four-dimensional vector. Therefore, we aim to build a network that outputs two vectors: one representing the object's position with respect to the camera reference frame and one indicating its orientation. Regarding the data that we use to feed the network, we have at our disposal the frames from a depth camera that provides a color image stream along with the per-pixel depth information for each frame.

The general structure of the network is illustrated in Figure 3. It can be divided into three almost independent blocks: a preprocessing segment that localizes in the color frame the object for which we want to estimate the pose, a center regression branch (called **TNet**) that estimates the three-dimensional coordinates of the object center and finally a pose estimation model (called **RNet**) that predicts the object rotation. Spitting the computation of the object translation from the rotation estimation generates a more flexible architecture where each task can be studied, trained and optimized separately. Moreover, it gives the opportunity to the user to apply only part of the whole model if the rest is not needed.

Our implementation is based on a per-pixel approach. The main idea is that the network selects a few pixels from both color and depth images and processes them separately to extract two vectors of geometric and color features from each pixel. The features are fused together to derive a pose prediction from each pixel. The final prediction is then selected as the pixel with highest confidence.

4.1. Instance segmentation

The instance segmentation network is the first component of our pipeline, being a preprocessing tool for the model. Its purpose is to locate an object in the image. In this way, we can feed the rest of the network with a cut out image that contains only the information regarding the object so that we can isolate it from the environment. For this part, we employ an instance segmentation network that predicts which pixels of the image belong to the object. The network that we select for image segmentation is Mask R-CNN [2] developed by Facebook AI Research in 2016.

4.2. Center regression - TNet

For this task we only exploit the depth image that has been cut out using the output mask

from the instance segmentation model. We extract a fixed number N of pixels randomly selected from the depth frame and we use the camera intrinsic parameters to reconstruct the 3D coordinates corresponding to each pixel. In this way we obtain a sparse point cloud restricted to the visible object surface. Therefore, the input of the center regression model is a 3xN matrix. From this, the network performs a feature extraction of local and global geometric structure for each pixel and then it extracts an estimate of the object center by predicting for each point in the input its translation with respect to the center.
For training this network we need to define a loss function that computes the average prediction error of each pixel. Let x_i be the position of the i^{th} pixel, we define the distance between the center x_C as a vector Δx_i computed as follows:

$$\Delta x_i = x_C - x_i \quad for \quad i = 1, ..., N. \tag{1}$$

Then $\Delta \hat{x_i}$ represents the prediction of our model from the i^{th} pixel, while Δx_i is the target vector. So, the loss function of the translation can be defined as:

$$L^T = \frac{1}{N} \sum_{i=1}^{N} \|\Delta x_i - \Delta \hat{x_i}\|_1 . \tag{2}$$

4.3. Rotation prediction - RNet

The rotation prediction network is a more complex network that relies on both color and depth images to predict the rotation of the object. As for the center regression model, we randomly pick N pixels between the once selected by the instance segmentation pre-processing tool. We extract the color data from the RGB frame and the corresponding distance information from the depth frame for each pixel. These are fed to two different networks that compute the corresponding embeddings which are then fused together and processed by a final fully connected set of layers that derive from each pixel an estimate of the rotation in the form of quaternions.
For the task of extracting features from the depth information, we use the same structure as the initial layers of the TNet model. In this way we can reuse the weights learned for the center regression and the final model will share these layers. The feature extraction for the colored pixels is made with Resnet34 model, that is a neural network with 34 layers widely employed for learning the main attributes from an image. The extracted color and pointcloud embeddings are then combined and fed to a fusion module that derives its structure from the DenseFusion network. This part of the network concatenates each pair of features and generates a fixed-size global feature vector using an average pooling layer. At the end, this global feature vector is appended to the local feature vector for each pixel. In this way, we enrich each dense pixel-feature with the global fused feature to provide a global context. We feed each of the resulting per-pixel features into a final network that predicts the object's 6D pose.
To train this network we need to define a distance between the predicted and the target rotation. To this end, we define the per-pixel orientation error as follows:

$$L_i^R = 1 - |\langle\, q_{target},\, \hat{q_i}\,\rangle|. \tag{3}$$

where $\langle\, q_1,\, q_2\,\rangle$ denotes the inner product of the corresponding quaternions.
The overall loss can be defined as a weighted average of the per-pixel losses on the per-pixel confidence c_i with a regularization term, that penalizes predictions with small per-pixel confidences. We express the loss as:

$$L = \frac{1}{N} \sum_{i=1}^{N} (L_i c_i - \omega log(c_i)) \tag{4}$$

where N is the number of randomly sampled pixels from the RGB-D image and w is a balancing hyperparameter. Intuitively, low confidence would limit the impact of the corresponding per-pixel loss but at the same time would incur a high penalty from the second term.

4.4. 3D pose estimation – Full model

Incorporating the TNet module to the RNet, sharing the same point cloud feature extraction layers, produces a network that is able to fully estimate the object 6 DoF pose. In this way, we obtain a model that simultaneously computes the center translation and the rotation of the object. The full model can be trained from zero or can benefit from the pre-trained weights of TNet and RNet that together provide a well-suited initialization.
For training the complete model we need to define a loss function that takes into account the

error on the translation and the error on the rotation predictions at the same time. For this purpose, we refer to [5] that proposes a loss which has been extensively used in literature and exploits the 3D objects model for the computation. We define the pose estimation loss as the distance between the points sampled from the 3D objects model in the ground truth pose and the corresponding points from the same model transformed by the predicted pose. Specifically, the loss to minimize for the prediction per-pixel is defined as:

$$L_i = \frac{1}{M} \sum_{j=1}^{M} \left\| (Rx_j + t) - (\hat{R}_i x_j + \hat{t}_i) \right\|_2 \quad (5)$$

where x_j denotes the j^{th} point of the M randomly selected 3D points from the object's 3D model, R and t are to rotational matrix and the translation vector that define the ground truth pose, and $\hat{R}_i$ and $\hat{t}_i$ define the predicted pose generated from the i^{th} pixel. For the overall loss that combines all the per-pixel losses we can use the same function 4 defined for RNet.
The loss function presented above is only well-defined for asymmetric objects, where the object shape or texture determines a unique canonical frame. Symmetric objects have more than one and possibly an infinite number of equivalent canonical frames, which leads to ambiguous learning objectives. For this reason, we define an ad-hoc loss function for symmetric objects that computes the minimum error between all the possible equivalent canonical frames base on the object symmetries. We write the function as follows:

$$L_i = \min_{\underline{k} \in K} \frac{1}{M} \sum_{j=1}^{M} \left\| (Rx_j + t) - E_{\underline{k}}(\hat{R}_i x_k + \hat{t}_i) \right\|_2 \quad (6)$$

where $E_{\underline{k}}(x)$ represents the equivalent canonical frames of the object.

5. Synthetic dataset

Our model requires a large and accurately labelled dataset composed of at least 1000 images to achieve good performances. However, it may be time-consuming and expensive to collect and annotate such a large dataset. To solve this problem, we can replace a percentage of images in our training set with synthetic data generated by a 3D rendering tool. For this purpose, we use an open-source rendering software called BlenderProc [1]. It provides a set of functions to generate a scene where multiple objects can be positioned and take virtual pictures of this scene from different perspectives. Blenderproc provides a render both for color and depth images as well as an automatic generation of perfect segmentation masks.
To understand how our network performs with the use of synthetic data we prepare a series of experiments for estimating the 6DoF pose of a single object, each with a different composition of real and synthetic images. From the results summarized in Table 1, we derive that replacing up to 30% of the dataset with synthetic images does not limit the model accuracy, while reduces the number of real images to be annotated.

Real	Synth.	Train error	Test error
1500	0	0.0222	0.0288
1250	250	0.0233	0.0280
1000	500	0.0237	0.0289
750	750	0.0235	0.0302
500	1000	0.0233	0.0321
250	1250	0.0203	0.0337
0	1500	0.0193	0.0395

Table 1: Prediction error on train and test set using different compositions of real and synthetic data

6. Experimental results

We evaluate our solution using an empirical approach and then making a comparison with the state-of-the-art works. Given a predicted 6DoF pose $[\hat{R}, \hat{t}\,]$ and ground truth pose $[R, t\,]$ we define the prediction error in the general case that includes both symmetric and non-symmetric objects, as:

$$ADD = \min_{\underline{k} \in K} \frac{1}{M} \sum_{x \in \Theta} \left\| (Rx + t) - E_{\underline{k}}(\hat{R}x + \hat{t}\,) \right\|_2 \quad (7)$$

where x is a vertex of totally M vertexes on the 3D object mesh Θ.
We train our network to learn the position and

Item	DenseFusion	Our model
Ape	0.6663	0.8651
Benchvise	0.7982	0.8852
Camera	0.6607	0.7993
Can	0.8277	0.8632
Cat	0.8832	0.9104
Driller	0.7889	0.8636
Duck	0.6629	0.7950
Eggobox	*0.0056*	*0.9512*
Glue	*0.0348*	*0.9678*
Holepuncher	0.6079	0.7644
Iron	0.9019	0.9351
Lamp	0.8829	0.9056
Phone	0.8347	0.9576
Total	**0.8071**	**0.8769**

Table 2: Percentage of correctly predicted poses on Linemod test set of DenseFusion and our model's components (symmetric object in italic)

orientation of a drill and test the accuracy of our model in a real application. On the drill dataset we achieve an average prediction error smaller than 2 cm on the test set (Figure 4).
Subsequently, we measure the performance of our implementation confronted with DenseFusion results on Linemod dataset [3]. The Linemod dataset consists of 13 low-textured objects, and it is wildly adopted by state-of-the-art pose estimation models. We summarize the outcome of the comparison in Table 2. The results show that our full model trained with viewpoint loss significantly outperforms the DenseFusion network trained on the same number of epochs.

7. Conclusions

In this work we propose a 6 DoF object pose estimation network that uses RGB-D images to predict object position and orientation with respect to a reference frame. We design a flexible model that can learn to predict object's translation and rotation independently or else the full pose jointly. In this way the model can easily adapt to the user's needs. We introduce a novel approach that deals with symmetric objects. At the end, we have generated an accurate and adaptable network that outperforms the state-of-the-art models and that can be employed to simplify tasks such as robotic grasping.

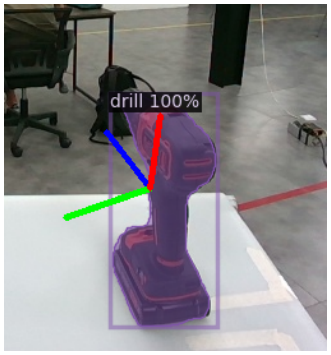

Figure 4: Example of drill pose detection from Intel D455 camera on robot chest

8. Acknowledgements

I would like to express my gratitude to Prof. Matteucci, both for introducing me to the fascinating world of neural networks and for its guidance in this thesis project. Many thanks also to Dr. Simone Mentasti for all the constant support and guidance on this journey. Thanks to Fabio Puglia and all the colleagues from Oversonic Robotics for giving me the opportunity to be a part of such an ambitious project.

References

[1] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam. Blenderproc. 2019.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. 2017.

[3] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. *2011 International Conference on Computer Vision*, pages 858–865, 2011.

[4] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. 2019.

[5] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2017.

[6] D. Xu, D. Anguelov, and A. Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. 2017.