RGBD Object Tracking for Visual Servoing

Rita Tlemcani
Stanford University
ritabt@stanford.edu

Abstract

Object tracking and visual servoing are crucial in vision-based robot control. Visual servoing allows a robot arm to autonomously grasp a target object. This paper aims at constructing an object tracking algorithm using a Convolutional Neural Network (CNN) trained on RGB-D data that will be used for visual servoing. After testing different Neural Network architectures, the model can classify images from a large RGB-D dataset containing 21 *labels with* 88.38% *accuracy. The model also uses a Bayesian Approximation method using Dropout layers in the CNN to measure the epistemic uncertainty of predictions which is the output uncertainty that comes from a lack in training data. The trained CNN can be used with the image pyramid and sliding window methods to draw a bounding box around objects that the model can classify with no uncertainty.*

1. Introduction

Object tracking has many useful applications towards autonomous robotics in addition to various fields such as video surveillance and editing. Autonomous cars rely on object detection and tracking to safely navigate their environment avoiding pedestrians and other vehicles. Robot-environment interaction can be achieved when a robot can localize objects in 3D space from sensors with no previous knowledge of the ground truth.

While much work has been done[9] to recognize object classes and bounding boxes from RGB data, some modern stereo cameras such as the Intel RealSense[6] have access to a fourth depth component which can provide more data to better train a CNN as well as provide more detailed information about the robot's 3D environment. In theory, the model can learn patterns in 3D shapes more analogous to how a human perceives their environment.

For *RGB-D Object Tracking for Visual Servoing*, the goal is to implement a program that can classify objects and detect their bounding boxes in real time from an off-the-shelf stereo RGB-D sensor for applications of visual servoing and robot-environment interaction. This is achieved through an object detection program using a simple model and trained on a dataset that doesn't provide ground truth bounding boxes.

Figure 1. Examples from the dataset. These images belong to the classes Apple, Ball, Camera, and Instant Noodles from left to right

This paper proposes a method to process RGB-D data for training and a simple CNN model that uses Bayesian Approximation to determine the uncertainty of the output. The trained CNN with uncertainty output can achieve better classification performance than a regular classification CNN in the object localization task.

2. Related Work

Real-time spatio-temporal feature tracking and object detection are key to building a visual servoing program, as explained by E. Marchand and F. Chaumette[12]. There are two main ways of tracking an object using visual input: feature-based and model-based tracking. The second approach allows for more robust visual servoing models and better performance because it is based on the 3D information of the model. Assuming that the 3D CAD model of the object of interest is provided, Marchand and Chaumette showed that it is possible to combine the 2D features and the object model for higher accuracy, efficiency, and stability in the visual servoing task.

YOLO[9] addressed the object detection problem using a novel and incredibly fast method. They used a single CNN pass, which explains the name You Only Look Once, during testing to determine the bounding boxes of objects in the image. They trained their network to be robust against background errors by training their model on full images. Their method is based on dividing the input image during test time into an $S \times S$ grid where each cell outputs a number

of bounding boxes along with the confidence score of each box. They use the Intersection Over Union (IOU) method using the ground truth bounding box to determine the performance of their model.

The IOU method has been commonly used as an evaluation metric for the accuracy of an object detection algorithm when the ground truth bounding boxes are part of the dataset. Y. Gal and Z. Ghahramani proposed[3] a simple way to estimate the model's uncertainty using Dropout layers. They proved that applying a Dropout layer before every weight layer is mathematically equivalent to a Bayesian approximation of the Gaussian process[10]. When using this method, it is important to keep the dropout layers active during testing and run the datapoints through the model multiple times. The distribution of the output labels can be used to determine the uncertainty of the model.

3. Dataset

The CNN is trained on the RGB-D Object Dataset from the University of Washington[11]. This dataset is one of the largest publicly available RGB-D datasets with many different labels. The dataset contains 51 categories (e.g Apple, Bowl, Calculator .. etc) and each category has between 3000 and 5000 data points. See Figure 1 for four examples from the dataset.

The dataset is unprocessed and contains more information than what is needed for this project. The preprocessing script generates images as numpy[4] arrays of shape (50, 50, 4) for each datapoint where the height and width are 50 and there are four channels for red, green, blue, and depth (RGB-D). The images in the dataset were crops of larger images. K. Lai et al.[11] manually cropped each image to fit the object of interest. Due to this, the datapoints have largely different shapes, which can be observed from the different sizes in Figure 1. The width of the images in the dataset ranges from 62 to 196 pixels and the height ranges from 47 to 156 pixels.

Because CNNs require that all the input datapoints are of the same size, the dataset needed to be modified to fit the size (50, 50, 4). The first approach was to pad the datapoints with 0 values accross all channels so that the minimum height and width are 50 and then take a random crop of size (50, 50, 4) from the padded image. This method created issues in training and testing because the CNN was trained on crops of the images only and was not trained on the full image and it was unable to predict a full image correctly. The second approach was to create a resize of each datapoint by first zero padding to create a square image and then resizing using OpenCV[1] helper functions to resize the image to (50, 50, 4). Using a random variable, 80% of the processed dataset was a resize and 20% was a crop using the first method mentioned above. The dataset was set up this way to prepare for image pyramids necessary to create bounding boxes (see *Methods* for more details). Training on crops helps the model recognize small parts from the object of interest.

Figure 2. Architecture Representation of the First Model

The images in the dataset are frames from a video taken of an object spinning on a turntable. As a result, continuous frames are almost identical. To reduce training time and to make sure that there are not repeated images in the dataset, one out of 8 datapoints were processed from 21 categories. After removing redundant datapoints and preprocessing each datapoint to unify the size, the dataset was separated into training, validation, and testing. There are in total $12,044$ datapoints and 80% was used for training, 10% for validation, and 10% for testing.

For easy interfacing with Pytorch[13], the dataset was divided into batches of size 100 and each datapoint was turned into a torch tensor of size (4, 50, 50). Also, the labels were made into one-hot vectors to work with the specific loss function used with the CNN model.

4. Methods

Two different CNNs were developed for this project. The first CNN architecture was for regular classification and the second CNN architecture used dropout as a Bayesian approximation[3] to measure the model's uncertainty. The first model converges faster while training and only needs one pass through the model during test time. The second model is slower to converge due to multiple Dropout layers. It is also slower during test time due to the fact that each datapoint needs 100 passes through the model during testing to determine how uncertain the model is about the classification of each datapoint. Although the second model is slower, it leads to better results in object detection. For object detection, image pyramid and sliding window technique were used to determine the location of an object without having access to the ground truth bounding box.

4.1. First CNN Model

The VGG model[15] showed that the use of many small filters (size 3x3) in the convolutional layers can result in high testing accuracy. By following the same logic, the

Figure 3. Architecture Representation of the Second Model

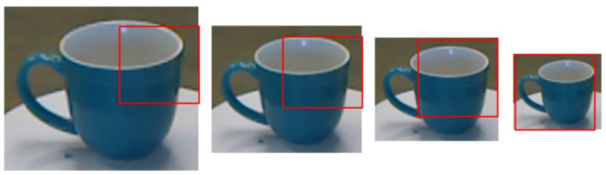

Figure 4. Representation of the image pyramid and sliding window techniques

model has 3 layers that use (3x3) filters then 2 layers with (5x5) filters in order to capture larger patterns from the image followed by a fully connected layer. Each convolutional layer was followed by a 2D batchnorm layer for regularization[7] and a ReLu activation layer. Batch Normalization also makes the model more robust to bad initialization and is a commonly used regularization technique. See Figure 2 for a visualization of the model's architecture.

The optimizer used for this model is a Stochastic Gradient Descent optimizer with momentum. The use of momentum with SGD has shown to allow for faster convergence in training[14]. The loss used is the multi class cross-entropy loss.

After training for 20 epochs (passes over the whole dataset) with learning rate 6×10^{-3}, momentum 0.3, and regularization strength 10^{-4}, the model was able to achieve 87.82% accuracy on the validation set (used for hyperparameter tuning) and 89.49% accuracy on the test set.

4.2. Second CNN Model

Athough the first model was able to achieve sufficient accuracy in classifying objects from the dataset, the model outputs a high confidence for a random label when introduced to objects not found in the dataset. Therefore, thresholding as a heuristic to remove unlearned objects and background noise would not work as intended.

To remedy this, Dropout as a Bayesian approximation[3] was used in the second model to find an approximation of the output's uncertainty. Inserting a dropout layer after each convolution layer is mathematically equivalent to a Bayesian approximation[10] of the Gaussian process. During testing, the dropout layers should be kept active and each datapoint is passed multiple times through the model. The mean of the output labels for each datapoint is considered the output label and the standard deviation represents the uncertainty of the label. This method gives an estimate of the epistemic uncertainty of the model which is the uncertainty that arises from a lack of training data.

The dropout layers take as an input a probability. The neurons in the following layer will be dropped by the input probability. Dropout is a commonly used regularization technique.

To implement this technique, the same convolutional layers from the previous model were used, but the Batchnorm layers were replaced with Dropout layers and a Softmax layer was added after the linear layer as proposed by Y. Gal et al.[3]. See Figure 3 for a representation of the model's architecture. The recommended[3] dropout probability of $p = 0.2$ was used as well as the recommended loss function and optimizer which are the Mean Squared Error loss (MSE) and the Adam optimizer.

Using a learning rate of 5×10^{-5}, a regularization strength of 10^{-4}, and after training for 100 epochs, the model was able to achieve 88.95% accuracy on the validation set and 88.38% accuracy on the test set.

4.3. Object Detection and Bounding Boxes

After training a CNN, we can use the image pyramid[2] and sliding window[8] methods to determine where an object is on an image and draw a bounding box around it.

4.3.1 Image Pyramid

The image pyramid method allows a model to be able to detect objects within an image at any scale[2]. This method works by taking in an original image and resizing the image multiple times. We then pass each one of the generated images through the model which creates a scale invariant classification model. See Figure 4 for a visualization of the image pyramid technique. It is important to keep track of the scale used at every level compared to the original image to be able to recover the original coordinates.

4.3.2 Sliding Window

The sliding window[8] technique works by taking multiple crops from an image and passing each one through the CNN. The crops are taken by starting at the top left corner and moving the window to the right and down at the end of the row using a stride value. A smaller stride value allows for more accurate object localization but a higher computational cost. As you can see from Figure 4, where the red rectangle represents the window, using the sliding window

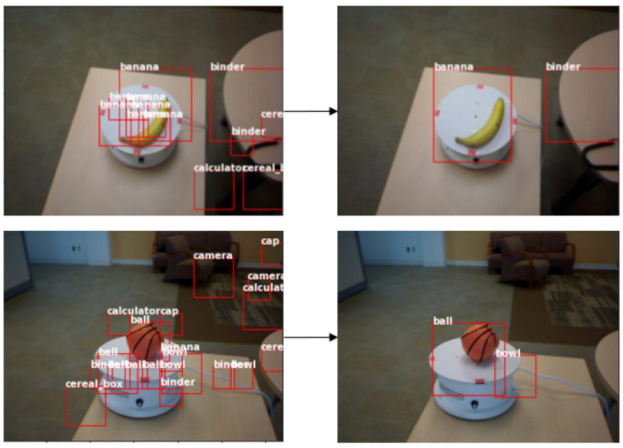

Figure 5. Before and After using the bounding box heuristic

with the image pyramid method allows the model to look at different scales of the input image.

5. Experiments

The RGB-D dataset offers two types of images: crop and scene. The crops are small images centered around the object of interest, while the scene is a larger image where at least one object from the cropped dataset is present (See Figure 5). The CNN was trained on the cropped images and the larger scene images were used to test the object detection algorithm. Since the dataset does not provide a ground truth bounding box for the objects, the scene images need to be inspected to determine if the bounding box is accurate.

The first model was trained on crops from the small images that only capture the object of interest so the dataset can have a unified size across all data points for the CNN model. The model showed a test accuracy of 74.69% when only training on parts of each image. When using the model for object detection, it recognized any curved edge (e.g the turntable) as categories that have a curved edge such as banana or bowl with high probability (between 60% and 80%). Because the model was not given full windows containing the objects as training data, the entire structure of an object is unknown which can lead to false positives. For example if trained on a window containing the edge of a white ceramic bowl, at test time a background image of a portion of the turntable's edge may be incorrectly classified with high confidence.

To account for this, the dataset was reset to use resized full object images instead of random crops which resulted in a higher test accuracy of 89.49% and performed better on the scene images but still returned high confidence false positives from the background. The second model was introduced as a way to determine the uncertainty of the model's output. When using the Dropout Bayesian Approximation method, every data point was passed through the model 100 times during test time while keeping the dropout layers active. Then, the mean and standard deviation are calculated, if the standard deviation is not equal to zero it is assumed that the model is uncertain about the label and the bounding box is not drawn.

Using the second model showed a great improvement and there were less false positives from the background. However, the problem was not completely eliminated, so a heuristic was used to reduce the false positives from the background. The heuristic used is that if a bounding box of a label has no overlapping bounding box of the same label, then it is removed. That is because 4 images are generated from the image pyramid with a sliding window of size (50x50) with a stride of 40 and there are repeated windows of the same area that will be classified with the same label. This method removed noise from the bounding boxes and created a bounding box around the object of interest with very little error. Figure 5[5] shows how the heuristic works. The model detects a bowl where the turntable is because the color and the depth are similar. The edge of the table is detected as a binder due to a similarity in shape.

6. Conclusions

In this project, two different CNN architectures were built and trained on an RGB-D dataset: a regular classification CNN and a CNN that outputs the uncertainty using a Dropout Bayesian Estimation method. The second model showed better results for the object detection and bounding box problem. The first model detects false positives with a high confidence which is a problem that couldn't be solved with thresholding. The second algorithm uses a technique that trains with dropout layers and keeps the dropout layers active during test time. At test time, each datapoint is passed through the model 100 times and the mean and standard deviation are used to determine the uncertainty of the model.

7. Future Work

The accuracy of this model may be improved with an additional catch-all background object category for training containing background crops from the large scene images. This method could help get reduce false positives seen in Figure 5.

Additionally, RGB-D videos from the dataset can be used to draw bounding boxes over multiple frames and track objects over time. This can be achieved through a per-frame use of the model or by training using temporal information.

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 2

[2] J. R. Bergen P. J. Burt E. H. Adelson, C. H. Anderson and J. M. Ogden. Pyramid methods in image processing, 1984. RCA Engineer. 3

[3] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, 2016. University of Cambridge. 2, 3

[4] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020. 2

[5] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007. 4

[6] Intel. Intel® realsense™ depth camera d455. https://ark.intel.com/content/www/us/en/ark/products/205847/intel-realsense-depth-camera-d455.html,. 1

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. Google Inc. 3

[8] Junseong Bang Jinsu Lee1 and Seong-Il Yang. Object detection with sliding window in images including multiple similar objects, 2017. 3

[9] Ross Girshick Joseph Redmond, Santosh Divvala and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. https://pjreddie.com/darknet/yolo/. 1

[10] Michael Kana. Uncertainty in deep learning. how to measure?, 2020. https://towardsdatascience.com/my-deep-learning-model-says-sorry-i-dont-know-the-answer-that-s-absolutely-ok-50ffa562cb0b A hands-on tutorial on Bayesian estimation of epistemic and aleatoric uncertainty with Keras. Towards a social acceptance of AI. 2, 3

[11] Xiaofeng Ren Kevin Lai, Liefeng Bo and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset, 2011. In IEEE International Conference on Robotics and Automation (ICRA). 2

[12] Eric Marchand and Francois Chaumette. Feature tracking for visual servoing purposes, 2005. Robotics and Autonomous Systems, Elsevier, 52 (1), pp.53-70. ffinria-00351898f. 1

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 2

[14] Ning Qian. On the momentum term in gradient descent learning algorithms, 1999. Center for Neurobiology and Behavior Columbia University. 3

[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. Visual Geometry Group, Department of Engineering Science, University of Oxford. 2