Exploiting Trademark Databases for Robotic Object Fetching

Joshua Song[1] and Hanna Kurniawati[2]

***Abstract*— Service robots require the ability to recognize various household objects in order to carry out certain tasks, such as fetching an object for a person. Manually collecting information on all the objects a robot may encounter in a household is tedious and time-consuming; therefore this paper proposes the use of large-scale data from existing trademark databases. These databases contain logo images and a description of the goods and services the logo was registered under. For example, Pepsi is registered under soft drinks. We extend domain randomization in order to generate synthetic data to train a convolutional neural network logo detector, which outperformed previous logo detectors trained on synthetic data. We also provide a practical implementation for object fetching on a robot, which uses a Kinect and the logo detector to identify the object the human user requested. Tests on this robot indicate promising results, despite not using any real world photos for training.**

I. Introduction

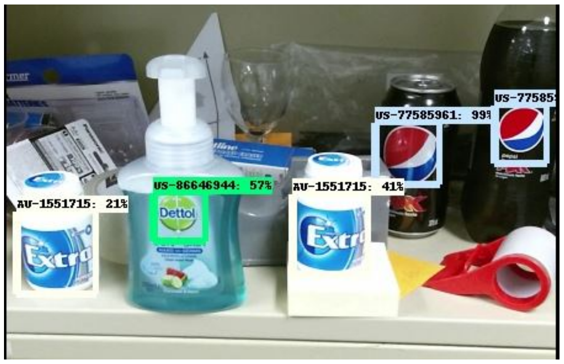
Object fetching, that is, getting a robot to recognize the object the user desires based on input from vision sensors and then retrieving it [1], [2], is an important task for service and assistive robots. These types of robots assist people at work or in their homes with daily tasks, which means to be effective, they must be able to recognize the large variety of objects in a household. However, such a recognition task remains an open problem. Convolutional Neural Networks (CNN) are advancing this domain tremendously, but require a huge amount of data. While category level (e.g. bottle, cup) classification is feasible, instance level (e.g. bottle of Pepsi or Coca-Cola, cup of Starbucks coffee) is more difficult since more data is required and the objects can be difficult to differentiate. There is work in progress on collecting 3D data for household objects, but this is a time consuming task and these datasets are currently incomplete [3], [4]. To alleviate the huge data requirement, we propose a novel approach: Exploit the availability of *structured data* from registered trademarks as a source of images and categorical information for both the CNN training and inference phase.

Companies place their logos on their products as distinguishing factors. Since companies would like to protect their logos, in general these logos are legally registered as trademarks. Organizations such as the World Intellectual Property Organization (WIPO) maintain databases of logos and their Nice Classification [5]. The Nice Classification identifies the goods and services category the logo was registered under. The list of goods is extensive and includes

This work was supported through an Australian Government Research Training Program Scholarship.

[1] School of ITEE, The University of Queensland, Australia.
[2] Research School of CS, The Australian National University.
`j.song@uq.edu.au, hanna.kurniawati@anu.edu.au`

(a) Movo needs to find a soft drink in the cluttered shelf



(b) Detections are shown with trademark registration numbers and detection confidence. Trademark US-77585961 (Pepsi) is registered under soft drinks and thus satisfies the request.

Fig. 1: Robot trial with logo detector trained only on synthetic RDSL images

categories such as “soft drinks” and “soap”. Therefore, such databases contain a massive amount of labelled data, i.e., images of the logos labelled with their brand owners and product categories, which could be used for training a CNN.

However, the characteristics of images in trademark databases are very different from the images that a robot must work with: The logos images in the databases are not placed on products, the images do not contain any other logos nor objects, and are obviously not taken by a (relatively cheap) camera. Considering these differences, if used *as is*, the images in the trademark databases are unlikely to help with object recognition in robotics.

This paper presents an approach for object fetching in robotics to benefit from the massive amount of labelled data available in the trademark databases, despite the aforementioned difficulties. In particular, we propose two uses of the trademark datasets. First is for object recognition. We propose an extension of domain randomization, called

Randomization-based Data Synthesizer for Logos (RDSL), that converts the trademark images of logos into synthetic "camera images". To this end, RDSL starts by automatically downloading the logo images from trademark databases and converting them into simple 3D scenes of logos (i.e., logos placed on simple geometrical objects). It then randomizes various parameters related to the logo placement, object placement, camera parameters, and background environment. The entire process of RDSL is automated. The synthetic images it generates are used as training data for a CNN for the object recognition component of an object fetching task.

The second proposed use of trademark databases is a simple method that exploits the categorical information to help the robot in understanding users' request. In particular, it will allow users to ask the robot to retrieve objects based on both categorical information (e.g., soft drinks or soap) or brands (e.g., Coke or Pepsi).

We test RDSL on a benchmark problem for logo detection (i.e., FlickrLogos-32 [6]) and test the entire object fetching pipeline on a Kinova Movo robot. The results are promising. The logo detectors that were trained using only synthetic data generated by RDSL outperformed previous logo detectors trained on synthetic data. We also demonstrate that the logo detector trained using only data generated by RDSL can be successfully used for object fetching tasks.

II. Related Work

Early approaches to logo detection used keypoint detectors, such as SIFT [7], [8]. More recent work has shown that CNNs outperform previous methods for logo recognition [9], [10], [11]. The improvements in computer vision have been employed by several companies in order to provide query-by-image services for logos. For example, TrademarkVision [12] allows a user to upload a logo image in order to find similar logos for the purpose of finding copyright infringement.

The CNN architectures referenced in this paper include VGG [13] and MobileNet [14]. MobileNet is a smaller and faster network while still having similar accuracy to VGG.

In a classification configuration, CNNs are invariant with regard to the position of the object within the image. However, it is often useful to localize and identify multiple objects within the image; this is termed object detection. One approach to this is Regions with CNN features (R-CNN) [15], which first finds regions of interest based on color and texture, then classifies the regions through a CNN and Support Vector Machine (SVM). Detection speed was improved with Fast R-CNN [16] and Faster R-CNN [17]. Faster R-CNN uses convolution features for both region proposals and class prediction. These convolution features may be taken from classification CNNs. Example configurations include Faster R-CNN with ResNet, Faster R-CNN with Inception and so on. An even faster object detection method (though slightly less accurate) is Single Shot MultiBox Detector (SSD) [18], which eliminates proposal generation and performs all computation in a single network.

In addition to class labels, object detectors also require their training data to be annotated with either bounding boxes or pixel masks for each object instance. Publicly available datasets include COCO [19] and Open Images [20], which are datasets for 91 and 600 object categories, respectively. Ammirato et al. [21] modified general object detectors trained on such datasets to be able to detect specific object instances (e.g. my mug instead of any mug), given several new images of the object instance.

Instead of training a new CNN model from scratch, the time and data required for training can be reduced through a process called transfer learning or fine-tuning [22]. In this process, a model that has been trained for a certain task is re-trained on data for the new task.

The training data can be expanded through augmentation techniques such as rotating, flipping and cropping images [23]. In addition to data augmentation, new training data can be generated through synthetic means. Eggert et al. [10] and Su et al. [11] generated synthetic data for logos by applying random geometric and illumination transforms to logos and pasting them on random photos. Montserrat et al. [24] followed a similar approach but used a CNN to estimate depth in the background image in order to blend the logo image more realistically.

Recent work [25] used simulators and domain randomization to generate synthetic data. The parameters of the simulator are randomized in unrealistic ways in order to force the CNN to learn the essential features of the target object. The synthetic data was then used to train an object detector for robot localization and grasping. They suggest that "with enough variability in the simulator, the real world may appear to the model as just another variation" [25]. Tremblay et al. [26] introduced *flying distractors* as a domain randomization component, which are random geometric shapes added to the scene. They then demonstrated the effectiveness of domain randomization for car detection.

Hinterstoisser et al. [27] generated synthetic data for a household objects detector by composing CAD models of those objects on random background images. They found that freezing the feature extractor layers and only fine-tuning the region proposal layers improved performance.

Similar to the above methods, our synthetic data generator, RDSL, also uses randomization. However, unlike the above methods that start with a relatively meaningful simulated scene (e.g., a table for [25] and road for [26]), in our problem, we do not have any such initial scene. In fact, the input images we have for the data generator are stand-alone logos, while in the real world, these logos are placed on various different objects.

III. Using Trademark Database for Fetching

Our pipeline for fetching tasks consists of two phases, a training phase and an execution phase. In the training phase, logo images are downloaded from the database and processed through our synthetic data generator (RDSL) described in Section III-A. The synthetic images are then used to train a Convolutional Neural Network (CNN) object detector. In the execution phase described in Section III-B, the robot uses the CNN to identify the object requested by the human user.

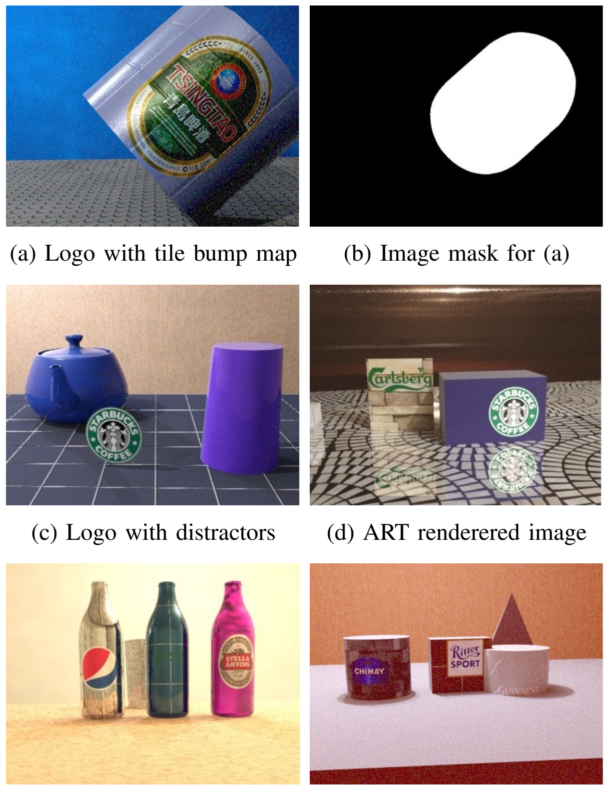
A. RDSL: A Method for Synthetic Data Generation

RDSL uses the logos from trademark databases to generate synthetic images that can be used to train a CNN logo detector for use in robotics applications. The entire process of this synthetic data generation is automated, therefore new logos can be generated with minimal human intervention.

RDSL starts by downloading the required logo images from trademark databases. These images are then preprocessed by removing the image's background, which for these images can be done easily using existing image editing tools, such as ImageMagick. The logo images were then superimposed on shapes such as cylinders, boxes and planes. The "UV maps" are specified for each shape to ensure that the 2D texture is correctly mapped in such a way that the logo is in the front of the object and hence visible to the camera.

To increase variability in the synthetic images, so as to improve the classifier's ability to generalize to real images, RDSL randomizes the following parameters:

- The number of objects of interest and distractor objects.
- Each object's properties: Color, texture, shape, materials (e.g., roughness and metalness), position, orientation, dimensions, and position of the logo on the object.
- Camera position and orientation.
- Lighting properties: Number, color, and brightness.
- Floor and wall materials.
- Additional noise, blurring, and coloring.



(a) Logo with tile bump map (b) Image mask for (a)

(c) Logo with distractors (d) ART renderered image

(e) Logos on bottle shapes (f) Noise and color shift

Fig. 2: Synthetic logo images generated through RDSL

Pseudocode 1 Synthetic data generation

```
function GENERATE MATERIAL
    Create a material from a random RGB value OR select a random texture
    Randomize material properties such as roughness and metalness
    Occasionally apply a bump map
return material
end function
for each selected logo do
    Download logo
    Remove logo's background
end for
function RDSL
    for the number of synthetic images required do
        for 1 to 3 randomly selected logos do
            Randomize logo size while maintaining aspect ratio
            $m \leftarrow$ GENERATE MATERIAL
            Composite logo on $m$
            Place $m$ on a random shape
            Randomize the shape's position, orientation and dimensions
        end for
        Randomize the camera position and have it point towards the logos
        for a random number of distractor objects do
            $m2 \leftarrow$ GENERATE MATERIAL
            Place $m2$ on a random shape
            Randomize the shape's position, orientation and dimensions while ensuring it does not block line of sight between camera and logo
        end for
        Randomize the number of lights and their color and brightness
        Randomize floor and wall materials
        Randomize noise, blurring, coloring
        Render image to file
        Render image mask to file
    end for
end function
```

Furthermore, RDSL applies rendering techniques such as bump mapping, which simulates bumps and wrinkles during lighting calculations. For example a tile bump map was used to generate Figure 2a. A few other examples of the synthetic images generated are shown in Figures 2c-f.

In addition to the logo class label, the training data also needs to be labeled with the pixel coordinates of a bounding box around the logo. This was done by performing an additional render of the logo alone as a separate mask image as shown in Figure 2b. It is then trivial to extract the bounding box from the mask image.

A summary of the steps followed to generate the synthetic images is given in Pseudocode 1. RDSL can use any available 3D modelling tools, such as Autodesk 3ds Max, Unreal Engine and Blender. It can also use any renderer.

B. Identifying Objects Through Logos

Our method is summarized in Figure 3. During the execution phase, the human user can either request for a particular brand, such as Pepsi or Pringles, or can make a request for a Nice Classification (NCL). The Nice Classification classifies the goods and services the logo is used for. Examples of Nice classes include soft drinks or potato chips. If the user made a brand request, the brand's Nice Classification can be looked up from the database. For the small number of logos used in this trial, the database can be stored onboard rather than needing to make a query over the Internet.

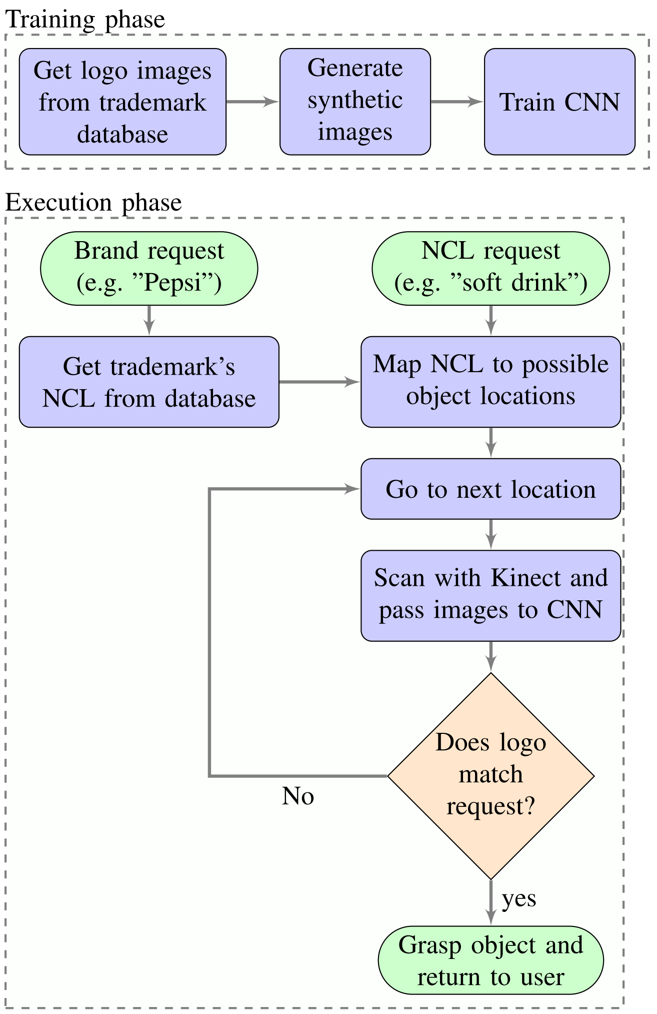

Training phase
Get logo images from trademark database
Generate synthetic images
Train CNN
Execution phase
Brand request (e.g. "Pepsi")
NCL request (e.g. "soft drink")
Get trademark's NCL from database
Map NCL to possible object locations
Go to next location
Scan with Kinect and pass images to CNN
Does logo match request?
No
yes
Grasp object and return to user

Fig. 3: Procedure for robotic object fetching with logos

In this work, we assume that the user will not provide any directions on where the target object is located. Instead, given a map of the environment and the target object's Nice Classification, the robot can narrow down the possible locations the object may be. For example, a household map can specify that food and drinks are located in the fridge, soap on a certain shelf, and so on.

The robot can then navigate to each location and perform a scan by panning the camera over the scene and passing the RGB frames to the object detection CNN. The CNN provides a bounding box of the logo in pixel coordinates, which must then be converted to world coordinates (i.e. XYZ coordinates in meters) in order to plan the grasping motion. This conversion can be done by using a distance measurement and the intrinsic and extrinsic matrices of the camera.

If the user made a brand name request, the exact matching logo must be returned. However, if the user made a Nice Classification request, then any logo under that classification can be returned.

IV. Experimental Results

The purpose of our experiments are two-fold. First, we compare the performance of the synthetic data generation method RDSL to existing synthetic data generation methods on benchmark test sets. Second, we test the applicability of our fetching pipeline (Section III) on a robotic platform and provide qualitative results.

A. Benchmark Tests

In this test, we used RDSL to generate 100 synthetic images for each logo in the FlickrLogos-32 data set. The logos were downloaded from TrademarkVision's database through their API. The randomization procedure of RDSL is implemented as a script inside Autodesk 3ds Max 2018. We limit the type of objects (both objects of interest and distractor objects in Pseudocode 1) to simple geometry, such as cylinders, boxes, etc. and only use textures from the Autodesk libraries for generating the synthetic data. The scanline renderer was the primary rendering engine used. While the Autodesk Raytracer (ART) renderer is able to produce more realistic images and is compatible with a wider range of 3ds Max materials, it is significantly slower to render and did not produce a noticeable effect on the classifier performance.

The CNNs were trained using the Tensorflow [28] framework. Two different object detection architectures were used, SSD with MobileNet and Faster RCNN with VGG16. SSD-MobileNet was trained for use in our robotics task where recognition needs to be carried out quickly and on a computationally limited platform. Faster RCNN-VGG16 was trained for benchmarking against previous work. The models were pre-trained on COCO and then fine-tuned for 200k steps on either the real photos from the FlickrLogos data set or the synthetic data generated through the process described in Section III-A. For comparison purposes, we also list the results as reported in [11], [24].

Training and benchmarking was done on a NVIDIA P100 GPU. The FlickrLogos [6] test set was used for benchmarking. It consists of 3,960 real world photos of the logos in various settings. As expected, SSD-MobileNet is faster, taking 198 seconds to completely classify the test set, compared to 918 seconds required by Faster RCNN-VGG16. Table I shows average precision values for each logo as well as the Mean Average Precision (mAP).

SSD-MobileNet trained on RDSL generated synthetic data outperforms previous work on logo detectors trained on synthetic data (SynImg-32Cls and SynthLogo). Moreover,

Title	Network Architecture	Training images per logo	Adidas Corona Google Ritt	Aldi DHL Guin Shell	Apple Erdi Hein Sing	Becks Esso HP Starb	BMW Fedex Milka Stel	Carls Ferra Nvid Texa	Chim Ford Paul Tsin	Coke Fost Pepsi Ups	mAP
RDSL (fast)	SSD MobileNet	**100 Syn**	44.4	49.4	19.5	62.0	80.8	49.0	43.3	22.1	**48.6**
			81.5	27.6	40.7	64.3	48.2	72.9	79.3	51.7	
			65.8	7.0	68.5	21.5	11.9	1.0	57.6	19.2	
			54.2	27.3	40.9	90.1	83.6	75.9	72.9	22.8	
Real (fast)	SSD MobileNet	40 Real	30.6	45.3	65.8	60.0	72.1	47.9	61.8	37.5	60.3
			92.7	56.1	40.1	88.7	63.4	83.6	79.2	72.3	
			77.6	85.0	61.9	24.3	36.1	31.4	84.5	29.6	
			58.4	37.0	69.8	91.1	41.3	73.7	72.2	59.5	
Real + RDSL Synthetic (fast)	SSD MobileNet	40 Real + 100 Syn	61.4	70.5	77.5	69.9	85.9	66.4	73.8	70.8	74.7
			99.3	53.8	59.5	92.7	76.6	86.7	87.3	86.3	
			79.9	85.6	78.3	53.3	48.0	50.4	95.5	47.1	
			76.9	37.1	95.3	97.0	85.2	89.4	81.7	71.8	
RDSL	Faster RCNN VGG16	100 Syn	38.8	55.3	35.8	18.8	59.4	15.5	28.7	26.8	42.9
			18.9	37.5	55.5	65.7	40.6	64.0	35.3	79.5	
			47.0	4.6	50.8	46.6	9.5	14.4	74.3	21.4	
			50.5	31.9	33.4	71.6	80.2	53.1	69.3	39.6	
Real	Faster RCNN VGG16	40 Real	52.3	81.6	77.6	69.9	79.4	66.3	81.8	71.0	78.6
			95.9	82.0	96.3	90.1	75.5	90.2	83.8	88.7	
			93.5	93.2	75.7	51.3	56.0	51.9	97.6	54.4	
			77.9	52.7	91.1	99.4	81.7	85.3	85.1	84.8	
Real + RDSL Synthetic	Faster RCNN VGG16	**40 Real + 100 Syn**	64.8	82.1	87.4	80.8	84.6	72.3	82.8	76.9	**82.0**
			96.9	82.9	90.4	86.9	77.5	88.6	87.2	90.6	
			95.6	96.2	72.6	63.9	50.4	58.9	98.7	65.9	
			80.5	53.7	95.2	99.0	92.9	89.6	87.4	89.6	
RealImg [11]	Faster RCNN	40 Real	68.1	79.1	84.5	72.3	86.4	68.0	78.0	73.3	81.1
			90.9	77.4	90.9	88.6	71.1	91.0	98.3	86.2	
			98.0	90.7	81.3	67.0	54.5	64.0	90.9	59.6	
			81.0	57.3	97.9	99.5	86.7	90.4	87.5	85.8	
SynImg-32Cls [11]	Faster RCNN	100 Syn	9.4	47.3	9.6	70.3	39.9	28.3	15.8	21.7	27.6
			6.1	11.1	4.1	44.7	22.9	60.9	43.6	28.8	
			23.0	16.7	43.1	9.9	4.6	1.1	38.1	9.7	
			22.7	38.3	15.5	65.6	28.7	55.1	27.4	20.1	
SynthLogo [24]	Faster RCNN VGG16	≈ 460 Syn	Not reported								47.7

TABLE I: FlickrLogos-32 test set (3,960 images) average precision benchmark results. The first six rows are results from our runs. The last three rows are results from the respective papers.

mixing real and our synthetic data resulted in higher performance than using real data alone. These results are in-line with the results from Su et al. [11] who found that mixing synthetic data and real data can improve performance if there is a shortage of real data.

When trained on real images, Faster RCNN-VGG16 was more precise than SSD-MobileNet by 5.2%. Surprisingly, Faster RCNN-VGG16 performs worse by 11% when trained on synthetic data. A possible explanation is that the smaller SSD-MobileNet network does not overfit to our synthetic data. We followed the method used in [27] for avoiding overfitting to synthetic data and first tried freezing the entire VGG16 feature extractor network, then freezing only the first convolution layer, but neither method improved precision.

Transfer learning (or domain adaptation) techniques, such as in [29], [30] have previously been shown to improve the accuracy of CNNs trained on synthetic images, but were not implemented due to two main considerations: we wished to compare our synthetic data (and not network architectures) to previous work, which did not employ these techniques, and test how well our simple strategy of using 0 real image for training applies to situations where images are noisy.

The precision appears to depends heavily on the logo. For SSD-MobileNet trained on our synthetic data, the Starbucks logo has an average precision of 90.1 compared to 19.2 for the Pepsi logo. This is also apparent for the detectors trained on real images. It is possible that the set of test images for certain logos are more difficult than others, or perhaps some logos are simply more recognizable to the detector.

We performed an ablation study where we disabled certain randomization features one at a time while keeping the other features constant and measured the effect on the classifier accuracy. Removing the random shapes for the logo resulted in a mean average precision decrease of 8.4, not having random background textures caused a decrease by 9.6, and not having bump maps caused a decrease by 1.2. The FlickrLogos-32 dataset contains many logos on bottles, therefore we experimented with having bottle shapes in our synthetic data, however this did not improve performance.

B. Robot Trials

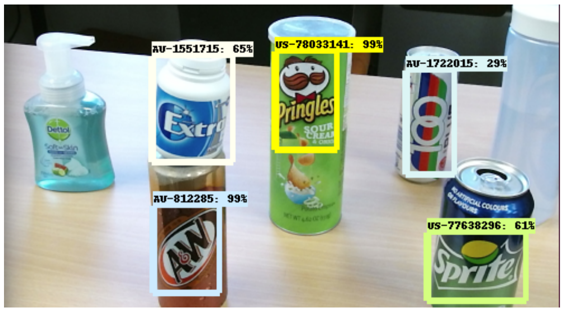
Movo [31] was used for performing a real-world object fetching trial. Movo is a mobile robotic platform equipped with two 6 DOF (degrees of freedom) arms, a Kinect and a 2D planar laser sensor. The Kinect for Xbox One is a time-of-flight sensor that provides both RGB images and distance measurement. Movo is also equipped with two Intel NUC5I7RYH computers, and comes ready to use with Robot Operating System (ROS) [32] and its motion planning framework MoveIt! [33].

To perform a scan for logos, Movo pans the Kinect over the scene and passes the RGB frames to the object detection CNN, which processes frames at a rate of 3 Hz. We trained SSD-MobileNet for the logos relevant to our scenario, which are trademarks that may appear in a typical household. The CNN was trained on 200 synthetic images per logo. We used an input image size of 600x600 instead of the default 300x300 as we found this performed better for small logos. The CNN provides a bounding box of the logo in pixel coordinates, which must then be converted to world coordinates. This conversion can be done by using the distance measurement and the intrinsic and extrinsic matrices. ROS's Kinect interface already performs this calculation in order to convert the depth image from the Kinect into a point cloud, which is a set of XYZ points. Therefore, we found it more convenient to extract the world coordinates by calculating the pixel's array index in the point cloud.

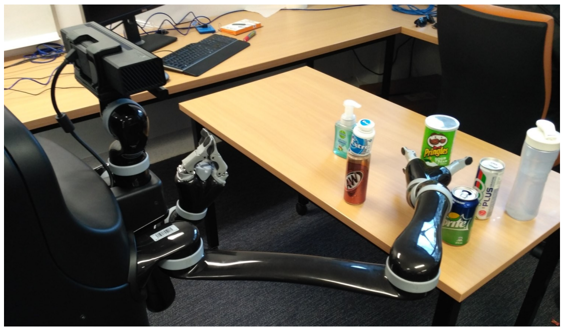
A scenario is shown in Figure 4. The products used here were soap (Dettol), chewing gum (Extra), potato crisps (Pringles), and soft drinks (Sprite, A&W, 100Plus). In this case, the user has asked Movo to bring food. We have placed the objects in such a way that the logos are facing Movo. If the logos were not visible, the objects can be detected by finding clusters in the point cloud and then rotated with the robot arm. After navigating to the table, Movo performed a scan. The detections from the logo detector are shown in Figure 4a with the trademark registration numbers. Unfortunately, the logo detector does struggle with small logos, in this case the Dettol logo. For each logo detected, Movo looked up the Nice Classification and then grasped Pringles, which is registered under class 29 (foodstuffs of animal or vegetable origin), and is also registered under "Snack foods, potato chips and potato crisps".

Our logo detector also works in visually cluttered scenes such as the one shown in Figure 1. Such scenes are often problematic for traditional, point cloud based object detection methods. This and other scenarios may be viewed in the accompanying video.

An issue that may cause confusion to the robot is that certain logos may be registered under certain Nice Classification while not having products for those classes. For example, the Mitsubishi logo is not only registered under vehicles, but also foods and drinks. We considered cross-checking the trademark database with Open Product Data [34], which is an open-source database of products, but found that database is currently quite incomplete.



(a) Movo needs to identify which object is food



(b) Movo grasps Pringles, which is trademarked under food

Fig. 4: Trial with logo detector trained on RDSL images

V. Conclusion

This paper investigated the use of trademarks to improve object recognition for fetching tasks by service robots. Rather than manually collecting training data on objects a robot may encounter, which is tedious and time-consuming, we leverage existing structured data from trademark databases. To this end, we propose an automatic data synthesizer, RDSL, that takes trademark images as its only input and generates synthetic labelled data for training a CNN-based logo detector. Furthermore, during execution, we leverage categorical information (i.e., Nice Classification) from the trademark databases to help the robot identify the object that the human user requested. We demonstrated the effectiveness of our approach via experiments on benchmark logo detection problems and a fetching task on the Kinova Movo mobile manipulator.

There is still much room for future work, including improving the quality of synthetic data generated, implementing transfer learning techniques to resolve the domain shift, and understanding the distribution for randomization that will be most suitable for robotics tasks. In this paper, all randomization used a uniform distribution. However, it is likely that other distributions may cover the span of real-world scenarios better. We hope the ideas and results presented here will encourage the exploration of other ways of alleviating a common issue in robotics: the need for a huge amount of relevant, labeled data.

Acknowledgment

We thank TrademarkVision for providing us with access to their database.

References

[1] H. Huttenrauch and K. S. Eklundh, "Fetch-and-carry with cero: observations from a long-term user study with a service robot," in *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*. IEEE, 2002, pp. 158–163.

[2] M. Mast, M. Burmester, B. Graf, F. Weisshardt, G. Arbeiter, M. Španěl, Z. Materna, P. Smrž, and G. Kronreif, "Design of the human-robot interaction for a semi-autonomous service robot to assist elderly people," in *Ambient Assisted Living*. Springer, 2015, pp. 15–29.

[3] A. Kasper, Z. Xue, and R. Dillmann, "The kit object models database: An object model database for object recognition, localization and manipulation in service robotics," *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 927–934, 2012.

[4] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 509–516.

[5] WIPO. (2018) Nice classification. [Online]. Available: http://www.wipo.int/classifications/nice/en/

[6] S. Romberg, L. G. Pueyo, R. Lienhart, and R. van Zwol, "Scalable logo recognition in real-world images," in *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ser. ICMR '11. New York, NY, USA: ACM, 2011, pp. 25:1–25:8. [Online]. Available: http://www.multimedia-computing.de/flickrlogos/

[7] A. D. Bagdanov, L. Ballan, M. Bertini, and A. Del Bimbo, "Trademark matching and retrieval in sports video databases," in *Proceedings of the international workshop on Workshop on multimedia information retrieval*. ACM, 2007, pp. 79–86.

[8] J. Kleban, X. Xie, and W.-Y. Ma, "Spatial pyramid mining for logo detection in natural scenes," in *Multimedia and Expo, 2008 IEEE International Conference on*. IEEE, 2008, pp. 1077–1080.

[9] S. Bianco, M. Buzzelli, D. Mazzini, and R. Schettini, "Deep learning for logo recognition," *Neurocomputing*, vol. 245, pp. 23–30, 2017.

[10] C. Eggert, A. Winschel, and R. Lienhart, "On the benefit of synthetic data for company logo detection," in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 1283–1286.

[11] H. Su, X. Zhu, and S. Gong, "Deep learning logo detection with data expansion by synthesising context," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 530–539.

[12] TrademarkVision. (2018) Trademarkvision. [Online]. Available: https://trademark.vision/

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[16] R. Girshick, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[20] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, S. Kamali, M. Malloci, J. Pont-Tuset, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, and K. Murphy, "Openimages: A public dataset for large-scale multi-label and multi-class image classification." *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

[21] P. Ammirato, C.-Y. Fu, M. Shvets, J. Kosecka, and A. C. Berg, "Target driven instance detection," *arXiv preprint arXiv:1803.04610*, 2018.

[22] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[23] J. Wang and L. Perez, "The effectiveness of data augmentation in image classification using deep learning," Technical report, Tech. Rep., 2017.

[24] D. M. Montserrat, Q. Lin, J. Allebach, and E. J. Delp, "Logo detection and recognition with synthetic images," *Electronic Imaging*, vol. 2018, no. 10, pp. 337–1, 2018.

[25] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 23–30.

[26] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," *arXiv preprint arXiv:1804.06516*, 2018.

[27] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," *arXiv preprint arXiv:1710.10710*, 2017.

[28] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[29] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim, "Image to image translation for domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4500–4509.

[30] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, "Domain adaptive faster r-cnn for object detection in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3339–3348.

[31] Kinova. (2018) Movo mobile manipulator. [Online]. Available: https://www.kinovarobotics.com/en/products/mobile-manipulators

[32] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[33] I. A. Sucan and S. Chitta. (2018) Moveit! [Online]. Available: http://moveit.ros.org

[34] Open Knowledge Labs. (2018) Open product data. [Online]. Available: https://product.okfn.org/