Proceedings of the ASME 2019 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2019
August, 18–21, 2019, Anaheim, CA, USA

IDETC2019/CIE-97393

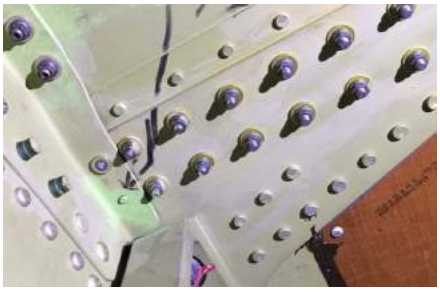
DOMAIN RANDOMIZATION FOR DETECTION AND POSITION ESTIMATION OF MULTIPLES OF A SINGLE OBJECT WITH APPLICATIONS TO LOCALIZING BOLTS ON STRUCTURES

Ezra Ameperosa [a,b,c], **Pranav A. Bhounsule** [b]
[a] Air Force Research Laboratory,
Materials and Manufacturing Directorate (AFRL/RX), Dayton, Ohio 43433
[b] Robotics and Motion Laboratory, Dept. of Mechanical Engineering,
The University of Texas San Antonio
One UTSA Circle, San Antonio, TX 78249, USA.
Emails: ezra.ameperosa@us.af.mil and pranav.bhounsule@utsa.edu
[c] The paper has been approved per # 88ABW-2019-0625.

ABSTRACT

Periodic replacement of fasteners such as bolts are an integral part of many structures (e.g., airplanes, cars, ships) and require periodic maintenance that may involve either their tightening or replacement. Current manual practices are time consuming and costly especially due to the large number of bolts. Thus, an automated method that is able to visually detect and localize bolt positions would be highly beneficial. In this paper, we demonstrate the use of deep neural network using domain randomization for detecting and localizing multiple bolts on a workpiece. In contrast to previous deep learning approaches that require training on real images, the use of domain randomization allows for all training to be done in simulation. The key idea here is to create a wide variety of computer generated synthetic images by varying the texture, color, camera position and orientation, distractor objects, and noise, and train the neural network on these images such that the neural network is robust to scene variability and hence provides accurate results when deployed on real images. Using domain randomization, we train two neural networks, a faster regional convolutional neural network for detecting the bolt and predicting a bounding box, and a regression convolutional neural network for estimating the x- and y-position of the bolt relative to the coordinates fixed to the workpiece. Our results indicate that in the best case we are able to detect bolts with 85% *accuracy and are able to predict the position of* 75% *of bolts within* 1.27 *cm. The novelty of this work is in the use of domain randomization to detect and localize: (1) multiples of a single object, and (2) small sized objects (*0.6 *cm* $\times$ 2.5 *cm).*

(a) Example application: Bolts on an aircraft fuselage



(b) Our experimental results

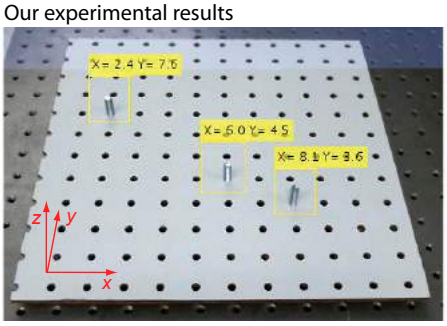


FIGURE 1. **Example application and our experimental results:** (a) A cracked aircraft fuselage has numerous bolts that need to be removed for maintenance operations [1]. (b) Our experimental results show identification and localization of three bolts on a workpiece. We obtain the results by training a neural network entirely in simulation. That is, no re-training is done on the real images.

1 INTRODUCTION

Fasteners such as bolts form an integral part of many structures (e.g., airplanes, cars, ships, bridges, machinery) and undergo maintenance cycles that involve either their tightening or replacement. Current practice involves visually detecting the bolts followed by manual tightening or measuring the coordinates using odometry (e.g., coordinate measuring machine, camera) and feeding these positions into the robot. Since there may be hundreds of such bolts, the manual approach is both time consuming and costly. An automated method that is able to visually detect and localize bolt position would be highly beneficial. In this paper, we demonstrate the use of domain randomization—a technique in deep neural network that enables simulation to real transfer of learned neural networks with no training on real images—to automate the process of detecting and localizing the position of multiple bolts in a workpiece.

2 BACKGROUND AND RELATED WORK

A problem similar to bolt localization is that of bolt-hole localization. Previous work in this area has used the circular geometry of the hole and used a circular hough transform based feature extraction [2,3] or template-based image matching [4] for localization. This process requires additional camera-workpiece calibration and image processing (e.g., segmentation), and a laser finder to detect the depth of the workpiece.

Furniture assembly robots such as the Ikeabot rely on the motion capture system for localization of the workpiece and then CAD models for the actual assembly [5]. However, the portability of a motion capture system and lack of accurate CAD models is a challenge in most bolt replacement applications. Another quick method is to manually train robot to disassemble the assembled piece. Then assuming that the external conditions are unaltered, simply reversing the robot commands allows assembling operations [6]. While this technique is good for assembly-line manufacturing (i.e., assembly in bulk), it is not efficient for maintenance operations where only a few bolts might need replacement.

Traditional image processing techniques involve some variant of template matching. Templates of the objects to be detected and localized are generated either using point clouds or using CAD models are generated offline and stored in memory. Then at run-time point clouds are obtained from the actual object, which needs to be localized. Then, features are extracted from the point cloud (e.g., using View Point Feature Histogram (VFH) [7]) and matched with the offline database. This method requires color and depth information (e.g., using either and RGB camera and range finder or RGB-D camera such as Microsoft Kinect [8]). It is also possible to use multiple RGB cameras to reconstruct the point cloud [9].

Convolutional neural networks provide an alternate method for localization. This involves training one or multiple neural networks on either real or computer generated images. For best performance, a substantially large dataset is needed. However, creating a large dataset of real images is costly as it requires recording data and manual labeling the data. Hence, it is preferable to use computer generated data for training the network. However, when the trained networks are used on real images, they lead to a poor performance. This is known as the reality-gap in robotics due to difference between computer generated images and real images. This gap has been traditionally tackled using an approach called domain adaptation and involves additional training of the neural network on real images [10, 11]. This vital step improves performance but comes at the cost of additional work needed to record experimental data and label it. Recently, Tobin et. al. [12] have demonstrated an alternate approach called domain randomization that works only with simulated data. The key idea is to train the neural network on synthetically generated images with a wide variety of textures, colors, light conditions, camera positions and orientations, noise, etc. The expectation is that with enough variability in the synthetic data, the real word will appear just as an instance of the synthetic images providing high precision, thus enabling easy sim-to-real transfer. Domain randomization has been used for pick and place among cluttered objects [12], robotic grasping [13], and classification of objects [14].

In this paper, we apply domain randomization for detecting and localizing bolts on a workpiece. Our work is adapted from that by Tobin et al. [12] but is novel in the following ways: (1) we detect multiple objects and locate their positions, and (2) the object of interest, bolts, are significantly smaller in size (0.6×2.5 cm). In particular, we use two neural networks; the first network detects multiples of the object from the world and the second one localizes the position of the bolt. The rest of the paper is organized as follows. We present the methods including metrics for evaluation in Sec. 3, the results in Sec. 4, the discussion in Sec. 5, and conclusions, including proposed future work, in Sec. 6.

3 METHODS

3.1 Overall approach

Our overall approach is depicted in Figure 2. We generate synthetic images of the workpiece with one or more bolts. We vary the lighting, color, texture, workpiece angle, and add distractor objects in the synthetic data set as shown in (a). This synthetic data set is exclusively used for training using the neural network described later in this section. The testing is done on real images obtained from a red-green-blue (RGB) camera as shown in (b). Note that real images are not in the training. This method of using only synthetic data with a lot of variability for training followed by evaluation on real images is known as domain randomization [12]. Figure 1 shows an example result of our algorithm detecting the three bolts on the workpiece and then estimating their position.

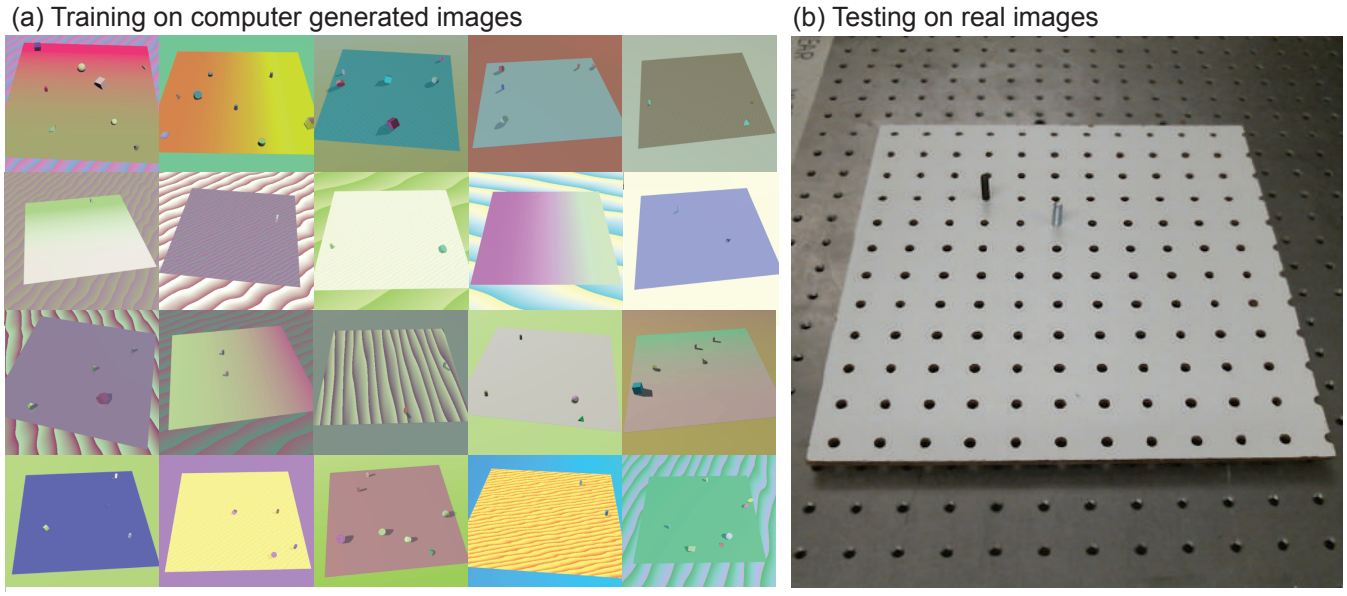


FIGURE 2. **Data for training and testing:** (a) Computer generated images are used for training the neural networks, (b) real image are used for testing. Note that real images were not used for training.

3.2 Computer generated images for training

We use the open source software Blender [15] to generate synthetic images for training. We place the workpiece at the center of the scene and its size is set to 30.5×30.5 cm. A table is set beneath the work piece that is arbitrarily sized to occupy space in the camera frame not covered by the workpiece. The bolts used in the synthetic images are taken from McMaster-Carr's online 3-D models [16]. To create data for training we create scenes resembling our physical system and include enough variability by randomizing the following factors: (1) the camera position and orientation; (2) lighting position and properties; (3) number, shape, and position of distractor objects; (4) number and position of the bolts (either 1 or 2 bolts) on the workpiece; and (5) color and texture of table, workpiece, bolts, and distractor objects. We generate scenes with these randomized features using Blender. We chose a resolution of 448×448. We create $75,000$ images (see Fig 2 (a)), with $50,000$ images containing only one bolt and $25,000$ images containing two bolts on the workpiece. We give more details about the factors that add variability to the data set in the following section.

3.2.1 Camera: In conventional vision-based localization, the camera needs to be calibrated with respect to the workpiece. This involves mapping the camera coordinates and axis to the workpiece coordinates and axis respectively and the image size with the that of the workpiece size. However, in our method, we bypass this step by varying the camera position and orientation as part of our training data. The rationale is that the neural network is able to automatically find the mapping between the image and workpiece given enough variability in the data. Another advantage of the step is that we can change the camera position and orientation within certain limits during testing without affecting the results.

We use a single camera in Blender to capture the synthetic images and randomize the camera's position with respect to the center of the workpiece. The camera is positioned $[-7.6, 7.6]$ cm from the center in the x-direction, $[-30.5, -45.7\ ]$ cm away in the y-direction, and $[30.5, 45.7\ ]$ cm above in the z-direction (see Fig. 1). After randomizing the position, we aim the camera to the center of the workpiece. We also randomize the camera Field of View in the range $[35^\circ, 40^\circ]$.

3.2.2 Lighting: We create two lighting sources; the first lighting source as the key lighting and the second source as backlighting. The key light is given a yellow tint while the backlight is given a light blue tint to mimic lighting in our actual testing environment. The backlighting is also used to soften shadows made by the key lighting. The key lighting's intensity is randomized, and the backlighting is approximately half of the intensity of the key lighting with some added variance.

The key light is positioned above the workpiece occupying a semi spherical space with a radius of 2.13 m from the center of the workpiece. The backlighting is positioned 180° about the center of the workpiece from the key lighting. Both light sources point at the center of the workpiece and add up to 10° variation.

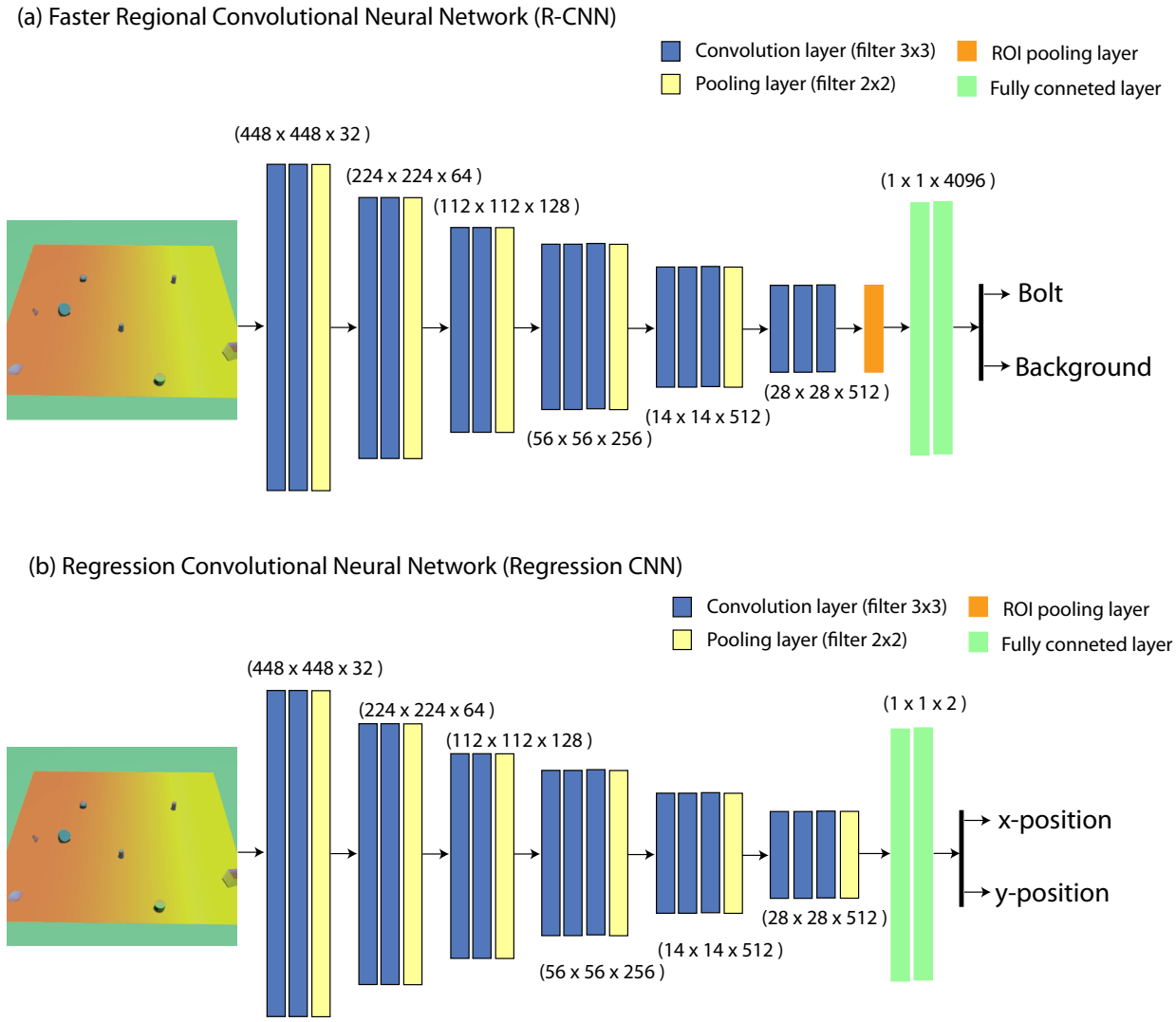


FIGURE 3. **The two neural networks used for detecting and localizing multiple bolts:** (a) A Faster Regional Convolutional Neural Network (Faster R-CNN) is used to classify multiple bolts, and (b) A Regression Convolutional Neural Network (regression CNN) is used to localize the position of the bolt.

3.2.3 Distraction objects: We use four shapes available in Blender as distractor objects: cubes, spheres, cylinders, and cones. The sizes of the distractors are made to be about the same size of the bolt. The number of distractors in each scene vary from 0 to 8 and are assigned a random shapes from the 4 shapes mentioned above. Also, the distractors are placed arbitrarily onto the workpiece.

The distractors act as a way to prevent the neural network from overfitting the synthetic data. Without the use of these distractor objects, the neural network would be vulnerable to false positive detections of random objects and any features not captured in training.

3.2.4 Bolts: We position one or two bolts anywhere on the workpiece but not on the workpiece edges because that would put the bolt outside the work area and we will not be able to measure the bolt position with accuracy (i.e., the ground truth). For each bolt, we setup the bounding boxes in the synthetic images using Blender and the cartesian coordinates (x- and y-position) of each bolt with respect to an origin fixed at the bottom left corner of the workpiece. The bounding boxes and x- and y-position were used as inputs to the neural network.

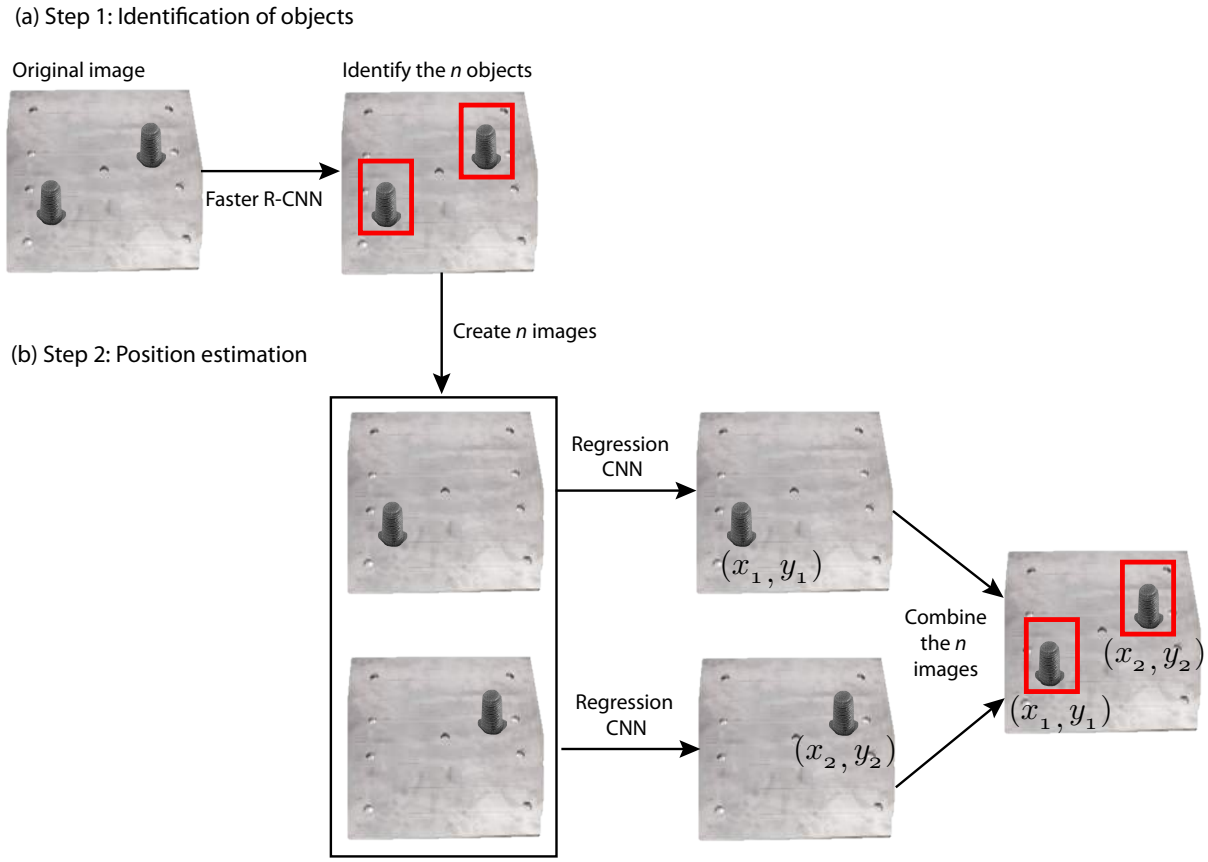


FIGURE 4. **Overview of our sequential method for object detection and position estimation:** (a) Step 1 involves identifying the n objects of interest and creating a bounding box for each of them using Faster R-CNN, and (b) step 2 involves creating n images, each with a single object followed estimation of the position of each object using Regression CNN.

3.2.5 Color and texture: We setup three textures for the bolt, distractor objects, table, and workpiece: solid color texture, marble texture, and a gradient texture. For each of the objects in the scene, we randomly set a texture to the objects and randomize the colors of the texture. We also add the possibility for objects to have 0% to 60% reflectivity.

3.3 Neural Networks

We use two neural networks: a Faster Regional Convolution Neural Network (Faster R-CNN) [17] for identifying objects and putting a bounding box around it and a regression Convolutional Neural Network (regression CNN) to estimate the x- and y-position of a single bolt. Both these are based on convolutional neural networks (CNN) that use filters to identify and learn features without using too many parameters. The Faster R-CNN incorporates a regional proposal network (RPN) in addition to a CNN that help to identify multiple objects as well as create a bounding box around them. For regression CNN, we add a regression layer to enable localization of the object. We use two instances of the VGG16 architecture with pre-trained weights from ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [18] for these networks. We also modify the networks to accept images of resolution 448 $\times$ 448 rather than 224 $\times$ 224. This is done to enable the use of a high resolution camera because of the small size of the bolts.

3.3.1 Training Faster R-CNN: We use VGG16 with pre-trained weights from ILSVRC. We modify the network to accept 448 $\times$ 448 images as mentioned earlier but also add two convolutional layers with ReLu activations following each layer. The convolutional layers have 32 filters with a filter size of 3 $\times$ 3. We add padding size of one and use a stride of one. We add a max-pooling layer after the second convolution layer with a pool size of two, zero padding, and a stride of two. We add these new layers to the beginning of the VGG16 architecture. We change the last fully connected layer to a two-neuron layer

and then change the softmax and classification layer to have an output label of either a bolt or background. Our network is shown in Fig. 3 (a).

We use the function *trainFasterRCNNObjectDetector* in MATLAB 2018a. We split the $75,000$ synthetic images (containing one or two bolts) into a training set of $67,500$ and validation set of $7,500$. We use the training set and the bounding boxes taken from Blender as input to the network. Stochastic gradient descent with momentum is used because MATLAB 2018a does not allow Adam optimization, which is generally used for Faster R-CNN. We use 0.9 for the momentum coefficient. We also add an L2 regularization with a coefficient of 10^{-4}. We use a mini batch size of 256 and train for three epochs. The training took about 1 day using graphics card GTX1080 on a standard desktop computer (circa 2017).

3.3.2 Training regression CNN: The regression CNN is shown in Fig. 3 (b) and is almost identical to the Faster R-CNN with only one change. The ROI pooling layer in Faster R-CNN (orange colored block just before the last two fully connected layers shown using green blocks in (a)) is replaced with a pooling layer (yellow block just before the two fully connected layers shown using green block in (b)). The fully connected layers here are used for regression and they output the x- and y-position of the bolt.

We use the function *train* in MATLAB 2018a. We split the $50,000$ synthetic images that contain one bolt into a training set of $49,000$ and a validation set of $1,000$. We do not use the images with two bolts ($25,000$ images) as the neural network in MATLAB can only estimate the position of one bolt at a time. We use Adam optimization with an initial learning rate of 10^{-4}, and a drop rate factor of 0.1 for every four epochs. We use a mini batch size of 32 and train for 9 epochs with no L2 regularization. The training took about 4 hours using graphics card GTX1080 on a standard desktop computer (circa 2017).

3.4 Sequential algorithm for testing on real images

Figure 4 shows the workflow used for testing the trained neural networks on real images. First, the image is passed through the Faster R-CNN to detect the bolts and put a bounding box on it as shown in (a). For each of the n bolts that are identified, we create n images by blending out all but one bolt as shown in (b). We then pass each of the n images individually through the Regression CNN to estimate the x- and y-position of each bolt. Finally, we combine all the images into a single image by putting a bounding box and labeling the position of each bolt.

3.5 Metrics

3.5.1 Metrics for identification: Our metrics for evaluation of the Faster R-CNN for identification are the "Recall", "Precision", and "F_1 Score". These are defined as follows.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{1}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

$$\text{F}_1\text{ Score} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{3}$$

Recall (Eqn. 1) is the ratio of correctly identified instances to all correct instances. It measures how well the detector is able to find all instances of the bolts. Precision (Eqn. 2) is the ratio of the correctly identified instances to total number of positive detections. This measure indicates how many detection are actually bolts among all the instances where the detector thought it was a bolt. With precision and recall we can better measure how well our detector performed in contrast to measuring just how many bolts it detected out of the total bolts. We use the precision and recall in finding the F_1 Score (Eqn. 3) at varying minimum acceptable confidence scores. The F_1 score is the harmonic mean of both recall and precision and is our performance score. An F_1 value close to 1 indicates a perfect recall and precision and gets worst as the value approaches 0. We use this metric to find the score for different confidence thresholds in detecting the bolt.

3.5.2 Metric for position estimation: The metric for evaluation of the regression CNN is the "Position error", defined as follows.

$$\text{Position Error} = \sqrt{(x-\hat{x})^2 + (y-\hat{y})^2} \tag{4}$$

where x and y are the true positions of the bolt, $\hat{x}$ and $\hat{y}$ are the position estimated by the regression CNN. Note that the origin for the measurement is the bottom left corner of the workpiece.

4 RESULTS

A video demonstrating the training and all code is given in Appendix A: Multimedia extension.

4.1 Real images for testing

Our workpiece is 30.5×30.5 cm wooden pegboard, which we place on an optical breadboard table. We create three test cases consisting of 20 scenes. The test cases are shown in Fig. 5. All test cases are similar in that they all have two bolts randomly placed on the pegboard but differ in the following ways. In test case 1, both bolts were stainless steel bolts. In test case 2, one bolt was a stainless steel bolt while the other was a black oxide

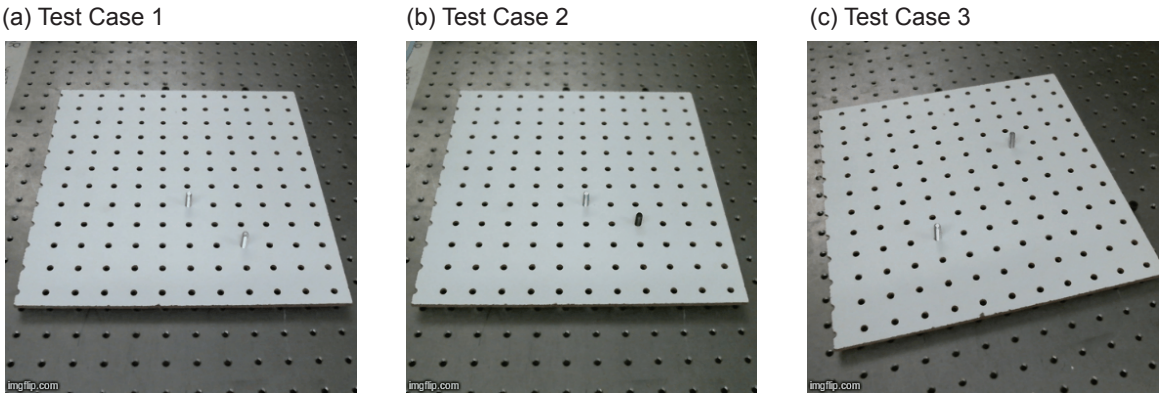


FIGURE 5. **The three test cases:** (a) Same color bolts, straight camera, (b) Different color bolts, straight camera, (c) Different color bolts, skew camera, different work piece colors.

bolt. In test case 3, we have a random combination of stainless steel, black oxide, and cadmium-plated bolt (looks golden in color). In addition for test case 3, we randomly chose either a white or a brown colored pegboard (the workpiece). Also, the camera position for all test cases was at an arbitrary distance and height but the camera was positioned straight ahead for test case 1 and test case 2 but it was 7.6 cm to the left for test case 3.

4.2 Results for identification

The F_1 score is plotted in Fig. 6 for the three test cases for detection confidence bounds between 50% and 6%. On average the F_1 scores for test cases 1, 2, and 3 are 0.75, 0.88, and 0.84 respectively. Note that F_1 scores are measures of false positives (see Eqn. 2) and false negative (see Eqn. 1). From these results we see that test case 2, which has bolts of different colors, has the highest F_1 scores and the test case 1, which has the same color bolts, has the lowest F_1 score. We observe that test case 1 and 3 both have about the same amount false positives, and they tend to appear in the same area throughout the sets: the bottom left part of the images, and the mid-right side of the images. We also observe that the detector has only failed in detecting bolts that are on the right side of the workpiece. However, we did not investigate the cause for these observed patterns. But we suspect that our domain randomization might not have covered certain combinations leading to poor performance in this part of the workpiece

4.3 Results for position estimation

The results for accuracy for the regression CNN for position estimation are shown in Fig. 7 and Table 1. The y-axis in the histogram indicates the percentage with respect to the number of bolts correctly identified, which are 35, 37, and 38 for the three test cases (see row 1 in Tab. 1) and the x-axis indicates the position error in cm (see Eqn. 4).

The results for test case 1 and test case 2 are almost similar as seen from Fig. 7 (a) (b) and Tab. 1 columns 3 and 4. The average errors for test case 1 and test case 2 were 1.22 cm and 1.07 cm respectively. That is, the test case 2 results are marginally better than that of test case 1. Note that the only difference between the test cases is that test case 1 uses similar bolts (steel bolt) while test case 2 uses different bolts (a steel bolt and a black oxide bolt). It is likely that the sharply different colors in test case 2 allows the neural network to do better at position estimation. On the other hand, the average error for test case 3 was 2.95 cm, about 3 times worse than test case 1 and test case 2. The main difference in test case 3 was the movement of the camera position to the right, thus indicating that the neural network is more sensitive to this parameter. We also notice that substantial errors in Fig. 7 (c) in the range $4-20$ cm. This is due to substantial high variability in this data set, which may not have been addressed adequately well in our synthetic data.

5 DISCUSSION

In this paper, we have demonstrated the use of domain randomization—a technique that relies exclusively on training on simulation followed by testing on real hardware—to identify and estimate the position of multiple bolts on the workpiece. Our approach consists of using the Faster Regional Convolutional Neural Network (Faster R-CNN) for object identification and laying down bounding boxes around the bolts followed by using a Regression Convolutional Neural Network (Regression CNN) for estimating the position of the bolts. We are able to detect between 88% to 97% of the bolts with an average error between 1.07 cm to 2.95 cm depending on the difficulty of the task.

Our work is novel in two aspects: (1) detection and estimation of position of multiple instances of the same object, and (2)

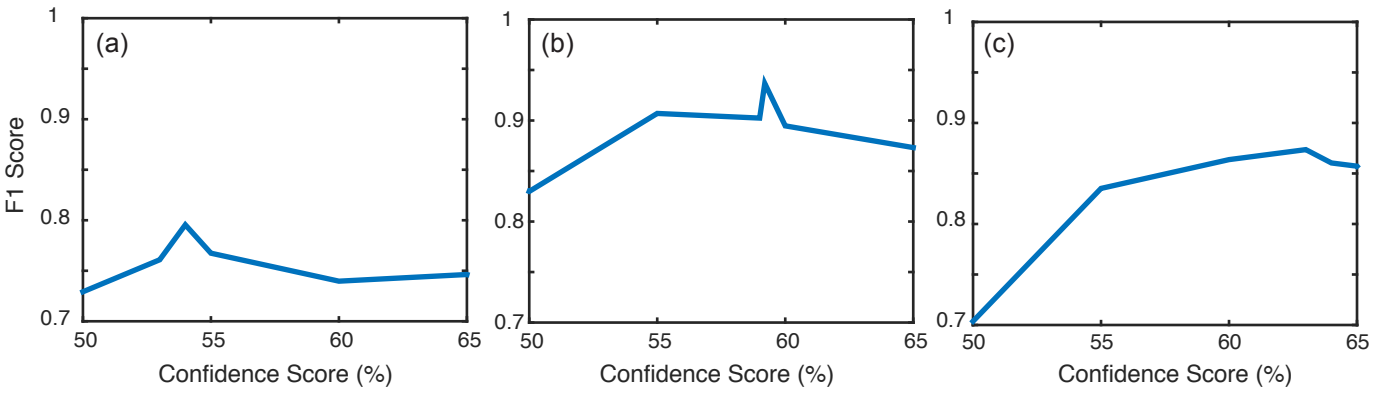


FIGURE 6. **F1 score for the three test cases** (a) Test case 1, (b) Test case 2, (c) Test case 3.

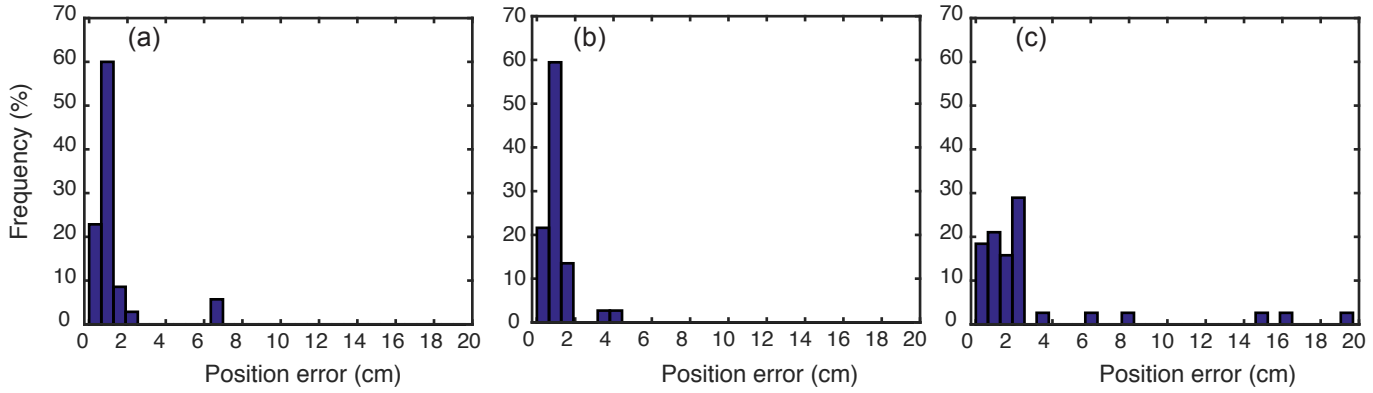


FIGURE 7. **Histogram for the three test cases as a function of the position error:** (a) Test case 1, (b) Test case 2, (c) Test case 3.

TABLE 1. Performance metrics for the three test sets to the two neural networks

Metric	Neural Network	Test Set 1	Test Set 2	Test Set 3
Bolts correctly identified	Faster R-CNN	35 out of 40	37 out of 40	38 out of 40
Average Error (cm)	Regression CNN	1.22	1.07	2.95
Standard Deviation (cm)	Regression CNN	1.35	0.74	4.39
Bolts with <1.27 cm or 0.5" error	Regression CNN	29 out of 35	30 out of 37	15 out of 38

detecting small objects, as each bolt is 0.6 cm in radius and 2.5 cm in length. Our work is based of Tobin et al. [12], who used domain randomization to detect spam objects among a series of randomly chosen objects. In their case the objects were big (we estimate 5 cm or more) and they were only detecting a single object in the frame. In spite of the smaller sized objects in our case, we were able to achieve similar position estimation accuracy, 1.5 cm. However, note that we only estimate the x-, y- position while Tobin et al. estimated the x-, y-, and z- position of the object.

A key aspect of our method is to use separate networks for identification (Faster R-CNN) and position estimation (Regression CNN). Although it takes more time to tune two networks independently, this has the advantage of being able to test and debug independently. For example, we may be able to retrain the network which is performing poorly rather than training a single network. Another feature is that the networks are used sequentially; first, Faster R-CNN detects the number of bolts and second, regression CNN determines the position using the results

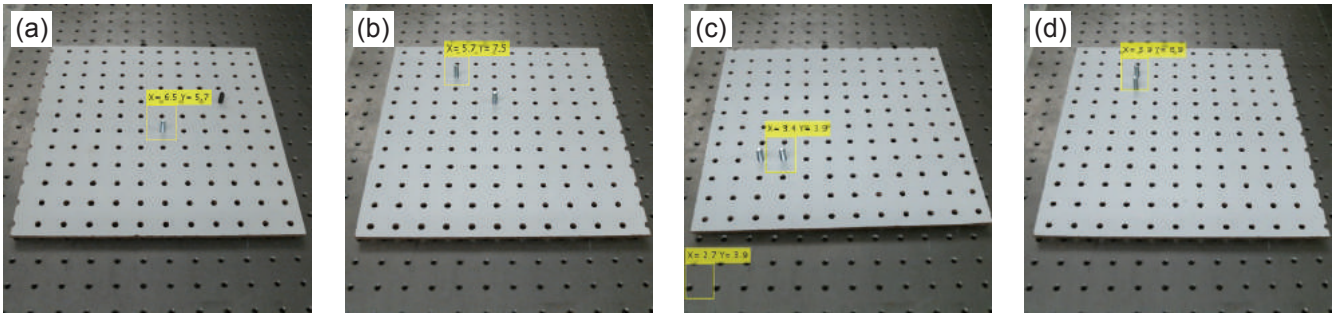


FIGURE 8. **Limitations of our method:** (a) Only one bolt detected, (b) only one bolt detected but with incorrect position, (c) only one bolt detected due to close proximity of the bolts, (d) two bolts identified as one by the bounding box.

from the Faster R-CNN output. The advantage of this sequential composition is that for training the regression CNN we only need to train on a single bolt, not multiples. However, a disadvantage of the sequential algorithm is that the results of the position estimation using Regression CNN depend on the results of the identification using Faster R-CNN. For example, in Fig. 8 (a) only one bolt is detected and thus only position can be estimated. Another example is shown in Fig. 8 (b) where although one bolt is detected, it has co-ordinates of the other bolt. This is explained as follows. With our method the algorithm would make two images, blurring out one bolt in one image then blurring out the other bolt in the second image, then send it through the position estimating network. However, in this instance, because the detector only identified one bolt, it does not blur any bolts and sends the unedited images through the second network. Through this we are not able to control which bolt the second network looks at to estimate its position. In this case we get the position of the undetected bolt and is unintentionally associated to the detected bolt.

Our algorithm has a difficult time localizing the bolts when they are close to each other as shown in Fig. 8 (c-d). In (c), the close proximity leads Faster R-CNN to identify only one bolt while in (d) two bolts are identified as a single one as indicated by the bounding box. Clearly, this problem can be eliminated if we had a closer view of the bolts. Another way would be to limit the bounding box size. One of the limitations of MATLAB 2018a was that we have reached the minimum ROI size for the bounding box and cannot reduce it further without possibly risking compromising the network.

We find that many of our limitations are to do with neural network toolbox in MATLAB 2018a. For example, MATLAB's minimum bounding box size does not allow us to identify multiple closely placed bolts. Another issue is that MATLAB does not allow simultaneous identification and regression and thus we had to use two different neural network independently for training. One way to do this would be to modify the RPN by adding more layers such that it does the bounding box regression loss to include a weighted x and y position regression loss. While MATLAB is continually improving its toolboxes, currently TensorFlow is a more mature and robust system for the image processing using neural networks. In hindsight, at the current time, TensorFlow seems to be a better choice for deep learning of images than MATLAB. Our choice for MATLAB was purely based on the availability of the neural network toolbox in our organization and the ease of using other toolboxes (e.g., computer vision for acquiring images).

6 CONCLUSION AND FUTURE WORK

This paper demonstrates that multiple objects of the same type maybe detected and localized leveraging state-of-the-art neural network classifiers and regressors and trained only in simulations using domain randomization. We believe that accuracy of 1.5 cm achieved through this method on hardware provides sufficient precision for robot grippers to pick and place tiny objects such as bolts (typically of size 0.6×2.5 cm).

There are multiple directions in which this research may be extended, as listed below:

1. Test the algorithm on additional bolts and conditions (e.g., distraction objects, lighting, camera position).
2. Combine the classification and regression networks into a single neural network and compare its performance with separate networks as done here.
3. Extend the regression to estimate the z-position as well as the orientation of the bolt on the workpiece.
4. Demonstrate robotic pick-and-place with objects localized using domain randomization.

ACKNOWLEDGMENT

We would like to thank Austin Deric for introducing us to domain randomization.

REFERENCES

[1] Australian_Government, 2018. Airbus a380-842 horizontal stabiliser structurebushing migrated. sdr 510020443. http://www.flightsafetyaustralia.com/2015/03/17-november-2014-19-january-2015-3/, July.

[2] Baek, G. R., Mo, Y. H., Jeong, J. S., Park, J. M., and Lim, M. T., 2011. "Cognition system of bolt hole using template matching". In 28th International Symposium on Automation and Robotics in Construction, ISARC 2011, Citeseer.

[3] Nagarajan, P., Perumaal, S. S., and Yogameena, B., 2016. "Vision based pose estimation of multiple peg-in-hole for robotic assembly". In International Conference on Computer Vision, Graphics, and Image processing, Springer, pp. 50–62.

[4] Choe, Y., Lee, H.-C., Kim, Y.-J., Hong, D.-H., Park, S.-S., and Lim, M.-T., 2009. "Vision-based estimation of bolt-hole location using circular hough transform". In ICCAS-SICE, 2009, IEEE, pp. 4821–4826.

[5] Knepper, R. A., Layton, T., Romanishin, J., and Rus, D., 2013. "Ikeabot: An autonomous multi-robot coordinated furniture assembly system". In Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, pp. 855–862.

[6] Miller, J. M., and Hoffman, R. L., 1989. "Automatic assembly planning with fasteners". In Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on, IEEE, pp. 69–74.

[7] Rusu, R. B., Bradski, G., Thibaux, R., and Hsu, J., 2010. "Fast 3d recognition and pose using the viewpoint feature histogram". In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, IEEE, pp. 2155–2162.

[8] Zhang, Z., 2012. "Microsoft kinect sensor and its effect". *IEEE multimedia,* ***19***(2), pp. 4–10.

[9] Collet, A., and Srinivasa, S. S., 2010. "Efficient multi-view object recognition and full pose estimation". In Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE, pp. 2050–2055.

[10] Gopalan, R., Li, R., and Chellappa, R., 2011. "Domain adaptation for object recognition: An unsupervised approach". In Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, pp. 999–1006.

[11] Matasci, G., Tuia, D., and Kanevski, M., 2012. "Svm-based boosting of active learning strategies for efficient domain adaptation". *IEEE J. Sel. Topics. Appl. Earth Observ. Remote Sens.,* ***5***(5), pp. 1335–1343.

[12] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P., 2017. "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, pp. 23–30.

[13] Tobin, J., Zaremba, W., and Abbeel, P., 2017. "Domain randomization and generative models for robotic grasping". *arXiv preprint arXiv:1710.06425*.

[14] Borrego, J., Dehban, A., Figueiredo, R., Moreno, P., Bernardino, A., and Santos-Victor, J., 2018. "Applying domain randomization to synthetic data for object category detection". *arXiv preprint arXiv:1807.09834*.

[15] Blender, 2018. Blender, a free and open-source 3d computer graphics software toolset. https://www.blender.org/, July.

[16] McMaster-Carr, 2018. Online catalog of mechanical, electrical, plumbing, and hardware parts. https://www.mcmaster.com/, July.

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

[18] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al., 2015. "Imagenet large scale visual recognition challenge". *International Journal of Computer Vision,* ***115***(3), pp. 211–252.

Appendix A: Multimedia Extension

1. A video of the hardware prototype available on this YouTube link: https://youtu.be/-DLU-bjDOhE.
2. The MATLAB and Blender code with test examples are on github: https://github.com/EzAme/DR2.