Proceedings of the ASME 2019 Dynamic Systems and Control Conference
DSCC 2019
October 8-11, 2019, Park City, Utah, USA

DSCC2019-8951

EXPERIMENTAL AUTONOMOUS DEEP LEARNING-BASED 3D PATH PLANNING FOR A 7-DOF ROBOT MANIPULATOR

Alex Bertino
Research Assistant
Dynamic Systems and Control Lab.
Dept. of Mechanical Eng.
San Diego State University
San Diego, California 92115
Email: abertino6245@sdsu.edu

Mostafa Bagheri
Senior Research Engineer
Dynamic Systems and Control Lab.
Dept. of Mechanical and Aero. Eng.
UC San Diego & San Diego State Univ.
La Jolla, California 92093
Email: mstfbagheri@ucsd.edu

Miroslav Krstić
Daniel L. Alspach Endowed Chair in
Dynamic Systems and Control
Dept. of Mechanical and Aero. Eng.
University of California, San Diego
La Jolla, California 92093
Email: krstic@ucsd.edu

Peiman Naseradinmousavi
Assistant Professor
Dynamic Systems and Control Lab.
Dept. of Mechanical Eng.
San Diego State University
San Diego, California 92115
Email: pnaseradinmousavi@sdsu.edu

ABSTRACT

In this paper, we examine the autonomous operation of a high-DOF robot manipulator. We investigate a pick-and-place task where the position and orientation of an object, an obstacle, and a target pad are initially unknown and need to be autonomously determined. In order to complete this task, we employ a combination of computer vision, deep learning, and control techniques. First, we locate the center of each item in two captured images utilizing HSV-based scanning. Second, we utilize stereo vision techniques to determine the 3D position of each item. Third, we implement a Convolutional Neural Network in order to determine the orientation of the object. Finally, we use the calculated 3D positions of each item to establish an obstacle avoidance trajectory lifting the object over the obstacle and onto the target pad. Through the results of our research, we demonstrate that our combination of techniques has minimal error, is capable of running in real-time, and is able to reliably perform the task. Thus, we demonstrate that through the combination of specialized autonomous techniques, generalization to a complex autonomous task is possible.

1 Introduction

Robot manipulators have many advantages over manual labor, they are more precise, consistent, faster, and stronger than even the most capable of humans [1]. Additionally, they do not suffer from exhaustion or lack of focus, and are capable of sustaining harsh environments. These traits are desirable in many environments that pose risks to humans, especially in industrial applications and medicine. However, the autonomous systems used to control such manipulators have historically had difficulties generalizing to complex tasks. This has led most manipulators into being used either for repetitive, easy to define tasks utilizing autonomous control, or more complex tasks under the direct control of a human operator. Furthermore, direct control via human input reduces the benefit of consistent performance,

since the human operator becomes the limiting factor of the system with respect to exhaustion and focus. Thus, the current limitations of autonomous systems represent a significant reduction in the potential of robot manipulators.

In recent years, many research efforts were carried out to improve the generalization ability of autonomous systems. These studies have been aided by significant advances in the fields of computer vision and machine learning. Such studies focused primarily on increasing the ability of autonomous systems to comprehend and act based on their environment, rather than relying on a predetermined state. Relevant to the purpose of this effort, research topics include object detection [2–5], pose determination [6–11], optimal grasp determination [11–20], and obstacle avoidance [21–27]. These studies all represent significant steps towards the autonomous operation of robot manipulators in complex tasks.

Object detection is a field of research useful in numerous applications, and has seen a significant amount of progress in recent years. Due to the research of He *et al.* [4], efficient object detection and pixel-wise segmentation is possible across thousands of object categories. Furthermore, its current speed of 5 fps means that object detection can now be performed in real-time, enhancing the ability of autonomous systems to react to a changing environment. This research is an iterative improvement of the earlier works of Girshick [2] and Ren *et al.* [3], each increased the performance and efficiency of the network.

Pose determination has seen a comparable amount of progress, due in part to the improvement of object detection networks and the increased availability of depth information in images. Early studies, such as that by Saxena *et al.* [6], sought to determine object pose by determining each of its 3 principal axes. More recent works, such as by Rad and Lepetit [7] instead sought to detect the corners of the object, and determined the pose through the well defined Perspective-n-Point (PnP) algorithm in computer vision. Research such as that of Pauwelset *et al.* [10] has shown that the determination of pose can be improved with depth information, collected either by stereo vision or with the use of RGB-D cameras. Additionally, Pauwelset *et al.* [10] demonstrated in their research that pose determination can be computed efficiently in real time, as their algorithm is capable of simultaneously detecting 150 objects at 40 fps.

Optimal grasp detection has been the focus of numerous studies in recent years, with many different methods explored. While some algorithms, such as Tremblay *et al.* [11], determined the optimal grasp purely through the determination of the pose, algorithms such as Lenz *et al.* [12] bypassed pose determination and compute the optimal grasp directly. Along with the presence or lack of depth information, studies vary in the conditions in which the grasp is to be performed. Johns *et al.* [18] investigated determining the optimal grasp accounting for uncertainty in the pose of the gripper, while Dogar *et al.* [17] studied optimally grasping an object while nudging others aside in a cluttered environment. The proper method to use for grasp determination is dependent on the type of problem the robot manipulator is intended to solve.

Obstacle avoidance, similarly to optimal grasp detection, is an active research topic with a large variety of tested methods. While most studies tend to focus on avoiding collision with the end effector, Yang *et al.* [22] investigated collision between an obstacle and the manipulator's other links, and avoiding collision via redundancy present in the manipulator. Among the studies that investigated collision with the end effector, there are still many differences present between studies. For instance, Wang *et al.* [26] proposed a non-linear model predictive controller for the obstacle avoidance problem, while Benzaoui *et al.* [24] presented a fuzzy adaptive control scheme, and Wei *et al.* [21] proposed an improved rapidly exploring random trees algorithm. Each algorithm has shown promise in the task of obstacle avoidance.

The goal of this paper is the autonomous development of an obstacle avoidance trajectory [28], combining computer vision, deep learning, and control principles. Specifically, our algorithm autonomously determines the 3D positions of an object, an obstacle, and a target pad, as well as determining the optimal grasp angle (orientation) to pick up the object. Once the object is picked up, a trajectory is autonomously determined to guide the object past the obstacle onto the target pad. Note that while most studies focus on a single aspect of autonomous systems, our effort examines the autonomous process as a whole. Key to this study is the examination of the performance of each step of the algorithm, and the interdependence of said steps. For this task, we utilize Baxter, a robot with two 7-DOF robotic arms and three cameras, as a case study.

The paper is organized as follows. In Section 2, we present a brief overview of the dynamics of Baxter's right arm, which will be used for object manipulation. In Section 3, we go over each step in the autonomous procedure. These steps are object detection via HSV-based scanning, distance determination via stereo vision, pose determination via deep learning, and the development of the obstacle avoidance trajectory. Finally in Section 4, we quantitatively asses the overall performance of the algorithm.

2 Mathematical Modeling

The redundant manipulator, which is being studied here, has 7-DOF as shown in Figs. 1 and 2. The Baxter manipulator's Denavit-Hartenberg parameters are shown in Table 1 provided by the manufacturer. The Euler-Lagrange equation leads to the robot's dynamic equations:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{1}$$

where, $q, \dot{q}, \ddot{q} \in \mathbb{R}^7$ are angles, angular velocities and angular accelerations of joints, respectively, and $\tau \in \mathbb{R}^7$ indicates the vector

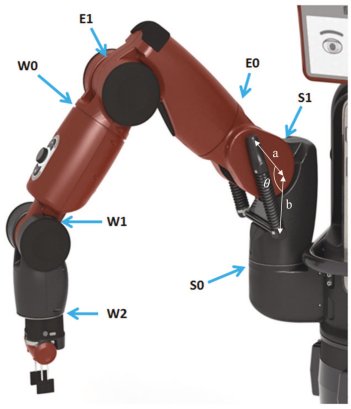
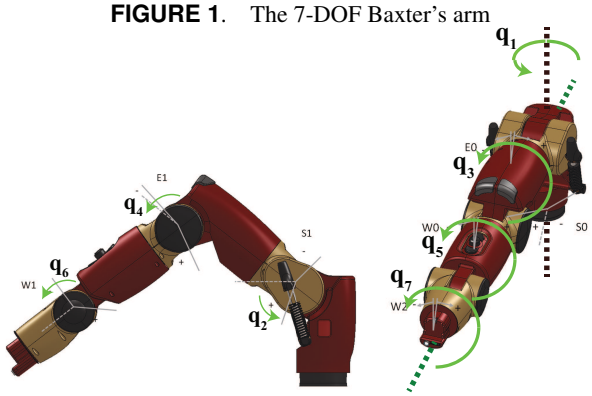


FIGURE 1. The 7-DOF Baxter's arm



(a) (b)

FIGURE 2. The joints' configuration: (a) sagittal view; (b) top view

of joints' driving torques. Also, $M(q) \in \mathbb{R}^{7\times7}$, $C(q,\dot{q}) \in \mathbb{R}^{7\times7}$, and $G(q) \in \mathbb{R}^7$ are the mass, Coriolis, and gravitational matrices, respectively. The coupled nonlinear dynamic model of the robot is verified in [29–32].

3 Fully Autonomous Operation

For the purpose of the autonomous task, the red object, blue obstacle, and green target mat coordinates are (x_p, y_p, z_p), (x_o, y_o, z_o), and (x_d, y_d, z_d), respectively. The origin of the coordinate system is the right corner of the table closest to Baxter. The x-axis is directed left along the table, the y-axis is directed upwards normal to the table, and the z-axis is directed away from Baxter along the table, as can be seen in Fig. 3. All positions are initially unknown and need to be autonomously determined.

To determine the 3D positions, Baxter's left-hand and head cameras each take a picture of the table from a predetermined position. Using the process of HSV-Based scanning, the centers of each of the three items are determined in 2D image frame coordinates. Once the image frame coordinates from each camera are determined, their corresponding 3D coordinates are determined utilizing stereo vision techniques.

TABLE 1. Baxter's Denavit-Hartenberg Parameters

Link	a_i	d_i	α_i	θ_i
1	0.069	0.27035	$-\pi/2$	θ_1
2	0	0	$\pi/2$	$\theta_2+\pi/2$
3	0.069	0.36435	$-\pi/2$	θ_3
4	0	0	$\pi/2$	θ_4
5	0.010	0.37429	$-\pi/2$	θ_5
6	0	0	$\pi/2$	θ_6
7	0	0.3945	0	θ_7

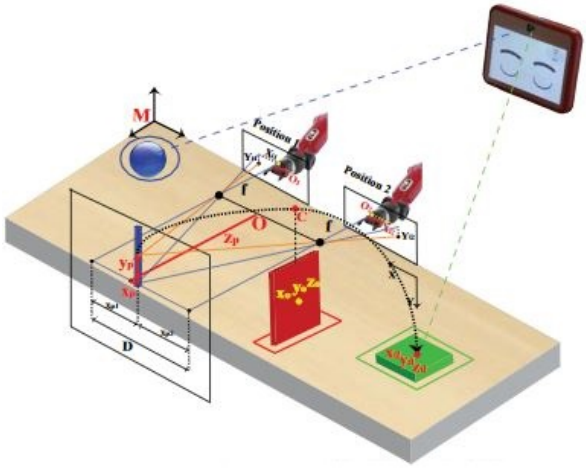
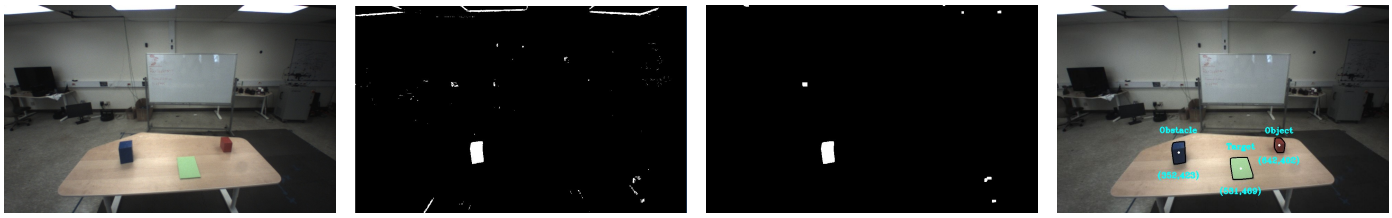


FIGURE 3. Physical layout of the autonomous task (in this picture object is blue and obstacle is red)

Once 3D positions are determined, Baxter's right manipulator is positioned in front of the red object. Utilizing a trained convolutional neural network (CNN), Baxter determines the 2D orientation of the object as the angle θ on the interval $0 \leq \theta \leq 180$. Once this angle is determined, the right manipulator is rotated by the calculated angle and the red object is picked up. Finally, using the determined positions of each object, a spline trajectory is calculated and performed moving the red object over the blue obstacle and onto the green target mat.

3.1 HSV-Based Scanning

In order to determine the position and orientation of each of the three items, it is first required to locate their position in a 2D image. As each of the items in the task has a unique distinguishing color, we use Hue, Saturation, and Value (HSV) based scanning to locate their 2D position. In the HSV color format, hue represents the "color" of the pixel, while saturation and value



(a) (b) (c) (d)

FIGURE 4. HSV-based scanning for object detection: (a) Original image; (b) Binary mask of "Blue" pixels; (c) Mask after morphological filtering; (d) Image with contours and centers drawn

measure how far the color is from "white" and "black", respectively. Expressing colors in HSV format allows the same color to possess a similar hue value under differing lighting conditions, which essential for color based tracking.

Once an image is converted from the RGB space (Fig. 4(a)) to the HSV space, the image is then scanned for pixels within a specified hue range, that possesses a minimum required saturation and value. The target hue value is based on the desired color to find, while the allowable hue range, the minimum saturation, and minimum value are all experimentally determined for each object color. All pixels than fall in the required range are set to white, while the other pixels are set to black. While the resulting binary mask, or black and white image, contains the desired object, the picture also contains noise, as can be seen in Fig. 4(b).

To reduce the noise present in the binary mask, morphological filtering is utilized [33]. The morphological filtering operations used in this effort are the opening operation to fill in holes in the object, followed by closing to remove background noise. These two operations are effective at removing noise in the binary mask, as can be clearly seen in Fig. 4(c). Small white streaks in the initial binary mask, caused by both the ceiling lights and blue tape on the floor, are almost completely removed from the mask, leaving the desired object as the most defined feature in the mask. After filtering is complete, a contour is drawn around the largest white object in the mask, and the center of the mask is used as the 2D location of the object in the image. As seen in Fig. 4(d), HSV-based scanning is effective as an object detection algorithm, and is able to generate accurate contours for the items.

3.2 Stereo Vision

Before 3D positions can be determined, it is first required to calibrate the stereo system. Unlike camera calibration, which estimates the intrinsic properties of the camera, stereo calibration is used to determine the homogeneous transform between two cameras.

During stereo calibration, multiple images of a large chessboard pattern are captured from the left-hand and head cameras, such that there are two pictures for every chessboard position, as shown in Fig. 5. The chessboard pattern is used due to the ease and accuracy of locating the corners between tiles in an image, which makes the pattern a good candidate for calibration. From each camera, 3D object coordinates (x,y,z) of each of the corners are known and their 2D image coordinates (u,v) are easily determined. Thus, we can use the PnP algorithm to determine the rotation and translation between the camera coordinate system and the chessboard. This algorithm is based on the relationship between 3D object coordinates and 2D image coordinates are given by the following equation

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2}$$

where f is the focal length of the camera, (c_x, c_y) is the center of the image, and R and T are the rotation matrix and translation vector, respectively, between the camera and the chessboard. s is a parameter chosen so that the third element of the homogeneous image coordinates is 1. Once the rotation and translation from the chessboard to each of the cameras are computed, the rotation and translation from the left-hand camera reference frame to the head camera reference frame can be determined as

$$R = R_2 R_1^T \tag{3}$$

$$T = -R_2 R_1^T + T_2 \tag{4}$$

where R_1 and T_1 are the rotation and translation, respectively, from the left-hand camera to the chessboard, R_2 and T_2 are the rotation and translation, respectively, from the head camera to the chessboard. Once R and T are determined, rectification of the images is possible.

Rectification is the process of virtually rotating stereo im-

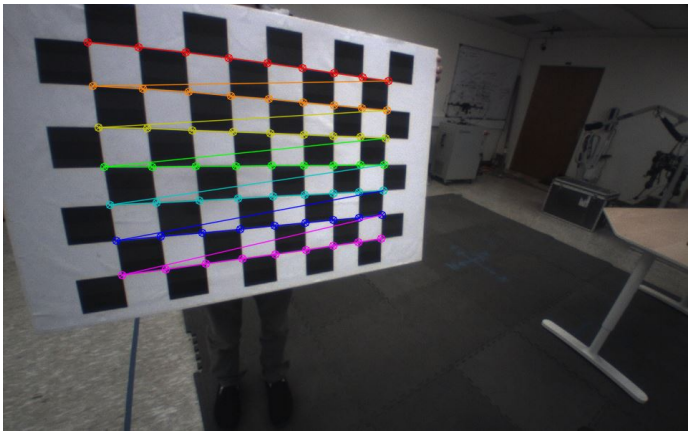


FIGURE 5. Stereo calibration: View from left-hand camera

ages, so that the images are horizontally rotated to each other on the same plane. The rectification algorithm uses the determined R and T, along with the intrinsic camera parameters, to create a map from each of the camera coordinate systems to the corresponding rectified system, as well as determining a horizontal shift T_x from the left rectified image to the head rectified image. In this effort, we utilize the rectification algorithm developed for the OpenCV library [34]. For the rest of this section, we refer to the rectified image coordinates $(u_1', v_1') = \text{map}_1(u_1, v_1)$ and $(u_2', v_2') = \text{map}_2(u_2, v_2)$ of each of the item centers.

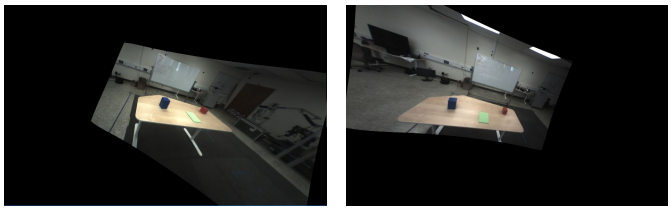
After the left-hand and head images are rectified (Fig. 6), we determine the 3D positions of the items using the following equation

$$s\begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & 0 \end{bmatrix}\begin{bmatrix} u_1' \\ v_1' \\ u_1' - u_2' \\ 1 \end{bmatrix} \tag{5}$$

where (x_L, y_L, z_L) are the 3D item coordinates relative to the left-hand camera, and s is a parameter chosen so that the fourth element of the homogeneous 3D coordinates is 1. Once the item coordinates in the 3D camera coordinate system (x_L, y_L, z_L) are determined, they are then rotated and translated into the table coordinate system (Fig. 7), described in the beginning of Section 3. The rotation and translation between the left-hand camera and the table were previously determined by applying the PnP algorithm to an image of the chessboard aligned to the table coordinate system.

3.3 Deep Learning

In order to determine the 2D orientation of the red object, we formulate and train a Convolutional Neural Network (CNN). Using a picture of the red object from the right-hand camera as input, our network predicts the angle θ on the interval $0 \leq \theta \leq 180$ between the table and the major axis of the red object, as shown in Fig. 8. To guide the training of this network, we define a cost function based on the mean square error between the predicted



(a) (b)

FIGURE 6. Stereo rectification: (a) View from left-hand camera; (b) View from head camera

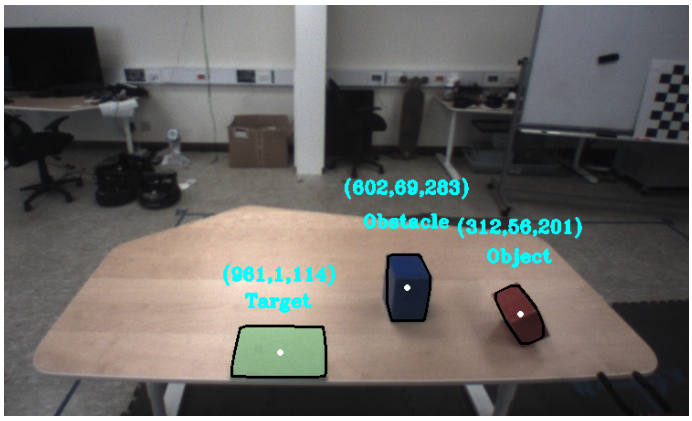


FIGURE 7. Calculated 3D positions (mm)

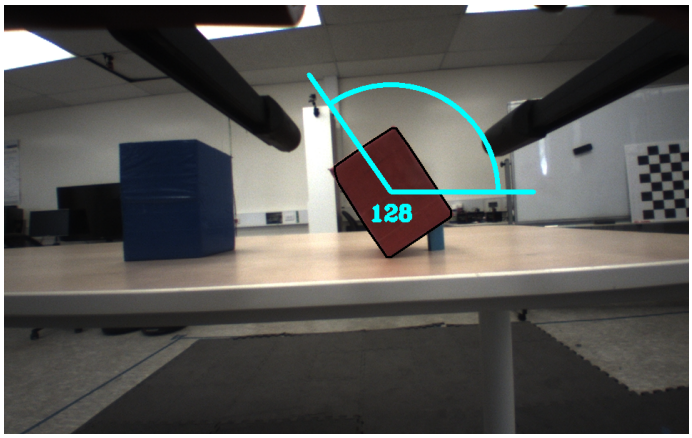


FIGURE 8. Orientation angle θ, predicted using deep learning

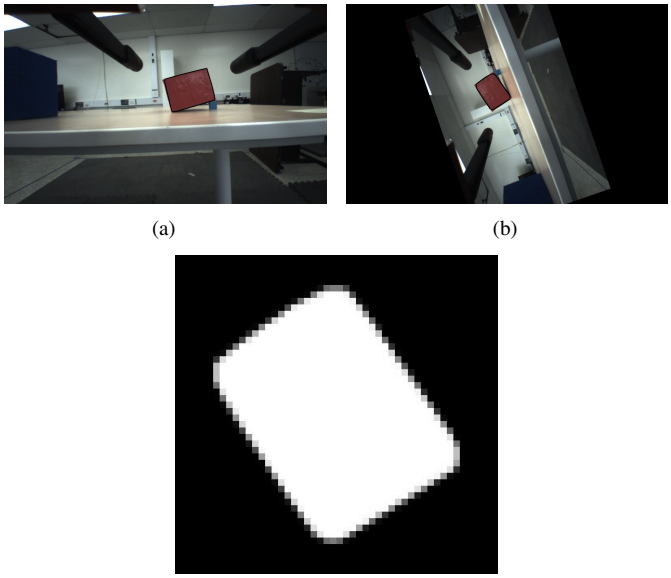
angle $\hat{\theta}$ and true angle θ

$$\text{MSE} = \frac{\sum_{i=1}^{N} \min\left\{\left|\theta-\hat{\theta}\right|, 180-\left|\theta-\hat{\theta}\right|\right\}^2}{N} \tag{6}$$

where N is the total number of samples being evaluated, and $\min\left\{\left|\theta-\hat{\theta}\right|, 180-\left|\theta-\hat{\theta}\right|\right\}$ is the shortest distance from θ to $\hat{\theta}$ compensating for a periodic output. We similarly define the mean error as

$$\text{ME} = \frac{\sum_{i=1}^{N} \min\left\{\left|\theta-\hat{\theta}\right|, 180-\left|\theta-\hat{\theta}\right|\right\}}{N} \tag{7}$$

which we use to quantitatively asses the performance of the CNN.



(a) (b)

(c)

FIGURE 9. Image preprocessing: (a) Initial right-hand image; (b) Right-hand image after random rotation; (c) Generated mask

In order to reduce the number of images required for training, as well as increase model performance, we preprocess (Fig. 9) each right-hand image before we pass it to the CNN. First, we apply HSV-based scanning to generate a mask representing the red object. We then crop the mask to the smallest square that inscribes the red object, in order to preserve the original aspect ratio. Finally, we resize the image to 48×48×1, and subtract the mean grayscale color value from the mask to prevent neuron saturation during training. This process reduces a 600×960×3 RGB image into a 48×48×1 image mask, reducing the number of input parameters by a factor of 750 while preserving the necessary data to predict the orientation.

The structure of our CNN is as follows. After the input mask passes through 6 alternating 3×3 convolutional and max pooling layers, the resulting data is flattened and passed through 6 dense layers of decreasing depth. We utilize the hyperbolic tangent as the activation function for all 3×3 convolutional and dense layers, excluding the final dense layer. For this layer, we instead use the remainder after division with 180. This insures that all predicted angles will be in the range $0^\circ \leq \theta \leq 180^\circ$ without saturation at 0° or 180°.

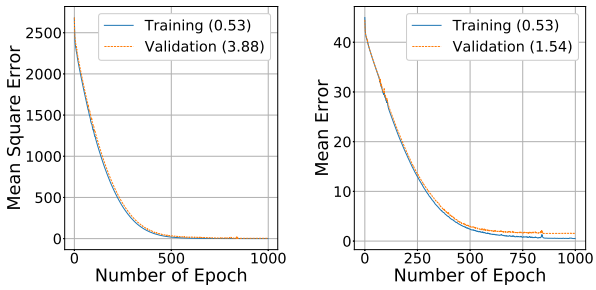
For the training of our network, we collected 140 images of the red object at different angles and distances from the camera, which we split into a training set of 120 images and a validation set of 20 images. To compensate for a small dataset, we augmented our data by applying 10 random rotations to each image, as seen in Fig. 9(b). Along with increasing the size of the dataset to 1400 images, this has the added benefit of increasing the spread of angles present in our data. The network was trained for 1000 epochs with a batch size of 200. We utilize the adamax optimizer developed by Kingma and Ba [35] with the suggested learning rate of 0.002.

3.4 Obstacle Avoidance Trajectory

Once 3D object position and the orientation of the red object, as well as the 3D position of blue obstacle and green target pad are determined, we formulate an obstacle avoidance trajectory comprising of the points $(x_{\text{obj}}, y_{\text{obj}}, z_{\text{obj}})$, $(x_{\text{obst}}, y_{\text{obst}} + K_1(h_{\text{obj}} + h_{\text{obst}}), z_{\text{obst}})$, and $(x_{\text{trg}}, h_{\text{trg}} + K_2 h_{\text{obj}}, z_{\text{trg}})$, where h_{obj} is the height of the red object, h_{obst} is the height of the blue obstacle, h_{trg} indicates the height of the green pad, and $K_1, K_2 \geq 0.5$ are spacing multipliers chosen to prevent collision. We employ a cubic spline between these three points, governed by the equations

$$\text{if} \quad t < \frac{t_{\text{opr}}}{2} \quad \begin{cases} y(t) = \alpha_1 t^3 + \beta_1 t^2 + \gamma_1 t + \delta_1 \\ x(t) = a_1 t^2 + b_1 t + c_1 \\ z(t) = a_1' t^2 + b_1' t + c_1' \end{cases} \tag{8}$$

$$\text{if} \quad t \geq \frac{t_{\text{opr}}}{2} \quad \begin{cases} y(t) = \alpha_2 t^3 + \beta_2 t^2 + \gamma_2 t + \delta_2 \\ x(t) = a_2 t^2 + b_2 t + c_2 \\ z(t) = a_2' t^2 + b_2' t + c_2' \end{cases} \tag{9}$$



(a) (b)

FIGURE 10. Convergence during training of CNN: (a) Mean Square Error; (b) Mean Error

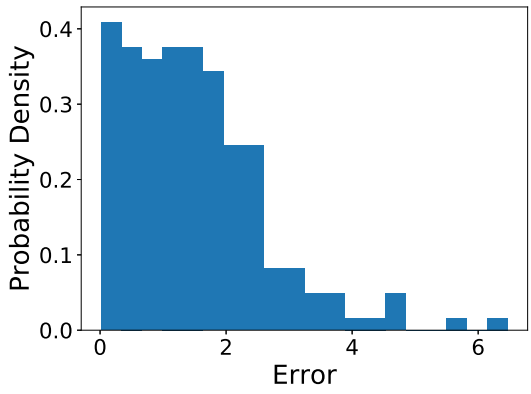


FIGURE 11. Histogram of orientation angle prediction error on testing dataset

where t_{opr} is the total operation time of the trajectory, and α, β, etc. are constants to be determined by applying the necessary boundary conditions. These conditions are as follows

$$P(t=0) = (x_{\text{obj}}, y_{\text{obj}}, z_{\text{obj}}) \tag{10}$$

$$P(t=\frac{t_{\text{opr}}}{2}) = (x_{\text{obst}}, y_{\text{obst}} + K_1(h_{\text{obj}} + h_{\text{obst}}), z_{\text{obst}}) \tag{11}$$

$$P(t=t_{\text{opr}}) = (x_{\text{trg}}, h_{\text{trg}} + K_2 h_{\text{obj}}, z_{\text{trg}}) \tag{12}$$

$$P'(t=0) = (0,0,0) \tag{13}$$

$$P'(t=t_{\text{opr}}) = (0,0,0) \tag{14}$$

$$y'(t=\frac{t_{\text{opr}}}{2}) = 0 \tag{15}$$

where $P = (x(t), y(t), z(t))$. Note that at time $t_{\text{opr}}/2$, the y derivative is zero, but the x and z derivatives are free.

4 Experimental Results

In order to determine the accuracy and precision of our 3D positions, we positioned the red object at 10 different locations on the table. At each position, we recorded the position calculated by our algorithm, and also determined the actual position. We determined that the mean distance between our calculated value and the actual value is 5 *mm*, with a standard deviation of 1.8 *mm*. This indicates our algorithm is capable of generating consistent results, which is crucial in order to properly pick up objects with a robotic manipulator. The satisfactory performance of our algorithm demonstrates that the combined use of image masks generated through object detection and stereo vision techniques is an effective method for determining 3D positions, and that the error carried over from each step of the algorithm did not prevent the algorithm from functioning as intended. Finally, this stage of the algorithm has an average run time of 78.4 *ms*, meaning 3D positions can be determined real-time with a potential 12.8 fps.

From Fig. 10, it can be seen that both the MSE and the ME exhibit smooth convergence throughout training. Also the validation error reaches a sufficiently small value of 1.54°. This value could reasonably be argued to be within the expected human error of our determination of the angle labels for training, which would indicate that the validation error converged to the minimum possible error. Similarly, in the testing set of 190 images after data augmentation, the ME is measured to be 1.49°. This indicates that the tuning of hyperparameters on the validation set did not cause overfitting when compared to the testing error. As seen in Fig. 11, this angle measure is also consistent, as the majority of the probability distribution, and thus the majority of predicted angles, are located within 4° of the actual angle value. Finally, this stage of the algorithm has an average run time of 17.2 *ms*, meaning that the orientation can be predicted in real-time with a potential 58.1 fps.

As a final performance test, we run the complete autonomous procedure on Baxter to complete the previously specified task. As can be seen in Fig. 12, Baxter is able to execute all stages of the procedure and complete the task. Each portion of the autonomous system operates on the accumulated error from previous steps in the algorithm, but these errors do not become large enough to prevent the system from functioning as intended. In the presence of negligible positional error from HSV-scanning and stereo vision, small angular error from the CNN, as well as fluctuations in the control of the right manipulator, the gripper is still successful in picking up the red object. Similarly, the manipulator successfully navigates the red object around the obstacle and on to the destination based on positions calculated earlier by the system. Despite the negligible accumulated error, algorithms developed to solve a specific autonomous problem can be effectively coupled in order to complete more complex tasks.

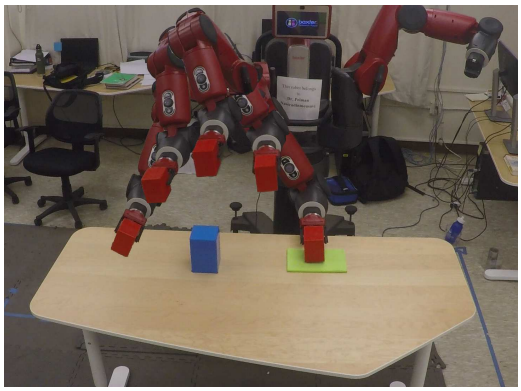


FIGURE 12. Experimental validation of complete autonomous procedure; see peimannm.sdsu.edu for the experiment AVI file

5 Conclusion

In this paper, we argued that robotic manipulators have not reached their full potential due to their current inability to perform complex tasks autonomously. We presented recent research efforts in a variety of topics relevant to the autonomous operation of robotic manipulators. Rather than focusing on a particular aspect relevant to autonomous operation, such as object detection or obstacle avoidance, we chose to analyze the performance of a complete autonomous system. We proposed a task involving object detection, pose determination, and obstacle avoidance. To execute this task, we developed an algorithm composed of multiple smaller procedures intended to solve a certain aspect of the combined task. These individual procedures employed a variety of computer vision, deep learning, and control techniques. Through the results of our research, we demonstrated that:

1. Each procedure in the algorithm operates with an acceptable error.
2. Each procedure in the algorithm is capable of running in real-time.
3. The complete algorithm is capable of reliably executing the task defined.

Thus, through the combination of specialized autonomous techniques, generalization to a complex autonomous task is possible.

Acknowledgment

This article is based upon work supported by the National Science Foundation under Award #1823951-1823983. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

[1] Bagheri, M., Ajoudani, A., Lee, J., Caldwell, D. G., and Tsagarakis, N. G., 2015. "Kinematic analysis and design considerations for optimal base frame arrangement of humanoid shoulders". In 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 2710–2715.

[2] Girshick, R., 2015. "Fast r-cnn". In Proceedings of the IEEE international conference on computer vision, pp. 1440–1448.

[3] 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.

[4] He, K., Gkioxari, G., Dollár, P., and Girshick, R., 2017. "Mask r-cnn". In Proceedings of the IEEE international conference on computer vision, pp. 2961–2969.

[5] Redmon, J., and Farhadi, A., 2018. "Yolov3: An incremental improvement". *arXiv preprint arXiv:1804.02767*.

[6] Saxena, A., Driemeyer, J., and Ng, A. Y., 2009. "Learning 3-d object orientation from images". In 2009 IEEE International Conference on Robotics and Automation, IEEE, pp. 794–800.

[7] Rad, M., and Lepetit, V., 2017. "Bb8: a scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth". In Proceedings of the IEEE International Conference on Computer Vision, pp. 3828–3836.

[8] Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D., 2017. "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes". *arXiv preprint arXiv:1711.00199*.

[9] Tekin, B., Sinha, S. N., and Fua, P., 2018. "Real-time seamless single shot 6d object pose prediction". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 292–301.

[10] Pauwels, K., Rubio, L., and Ros, E., 2016. "Real-time pose detection and tracking of hundreds of objects". *IEEE Transactions on Circuits and Systems for Video Technology,* ***26***(12), pp. 2200–2214.

[11] Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., Fox, D., and Birchfield, S., 2018. "Deep object pose estimation for semantic robotic grasping of household objects". *arXiv preprint arXiv:1809.10790*.

[12] Lenz, I., Lee, H., and Saxena, A., 2015. "Deep learning for detecting robotic grasps". *The International Journal of Robotics Research,* ***34***(4-5), pp. 705–724.

[13] Saxena, A., Driemeyer, J., Kearns, J., and Ng, A. Y., 2007. "Robotic grasping of novel objects". In Advances in neural information processing systems, pp. 1209–1216.

[14] Saxena, A., Driemeyer, J., and Ng, A. Y., 2008. "Robotic grasping of novel objects using vision". *The International Journal of Robotics Research,* ***27***(2), pp. 157–173.

[15] Saxena, A., Wong, L. L., and Ng, A. Y., 2008. "Learning grasp strategies with partial shape information.". In AAAI, Vol. 3, pp. 1491–1494.

[16] Jiang, Y., Moseson, S., and Saxena, A., 2011. "Efficient grasping from rgbd images: Learning using a new rectangle representation". In 2011 IEEE International Conference on Robotics and Automation, IEEE, pp. 3304–3311.

[17] Dogar, M., Hsiao, K., Ciocarlie, M., and Srinivasa, S., 2012. "Physics-based grasp planning through clutter".

[18] Johns, E., Leutenegger, S., and Davison, A. J., 2016. "Deep learning a grasp function for grasping under gripper pose uncertainty". In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 4461–4468.

[19] Pinto, L., and Gupta, A., 2016. "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours". In 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 3406–3413.

[20] Jang, E., Vijayanarasimhan, S., Pastor, P., Ibarz, J., and Levine, S., 2017. "End-to-end learning of semantic grasping". *arXiv preprint arXiv:1707.01932*.

[21] Wei, K., and Ren, B., 2018. "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved rrt algorithm". *Sensors,* ***18***(2), p. 571.

[22] Yang, C., Wang, X., Cheng, L., and Ma, H., 2017. "Neural-learning-based telerobot control with guaranteed performance". *IEEE Transactions on Cybernetics,* ***47***(10), pp. 3148–3159.

[23] Han, D., Nie, H., Chen, J., and Chen, M., 2018. "Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection". *Robotics and Computer-Integrated Manufacturing,* ***49***, pp. 98–104.

[24] Benzaoui, M., Chekireb, H., Tadjine, M., and Boulkroune, A., 2016. "Trajectory tracking with obstacle avoidance of redundant manipulator based on fuzzy inference systems". *Neurocomputing,* ***196***, pp. 23–30.

[25] Guo, D., and Li, K., 2016. "Acceleration-level obstacle-avoidance scheme for motion planning of redundant robot manipulators". In 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), IEEE, pp. 1313–1318.

[26] Wang, M., Luo, J., and Walter, U., 2016. "A non-linear model predictive controller with obstacle avoidance for a space robot". *Advances in Space Research,* ***57***(8), pp. 1737–1746.

[27] Lin, C.-J., Li, T.-H. S., Kuo, P.-H., and Wang, Y.-H., 2016. "Integrated particle swarm optimization algorithm based obstacle avoidance control design for home service robot". *Computers & Electrical Engineering,* ***56***, pp. 748–762.

[28] Bagheri, M., Krstić, M., and Naseradinmousavi, P., 2018. "Analytical and experimental predictor-based time delay control of baxter robot". In ASME 2018 Dynamic Systems and Control Conference, American Society of Mechanical Engineers, pp. V001T04A011–V001T04A011.

[29] Bagheri, M., and Naseradinmousavi, P., 2017. "Novel analytical and experimental trajectory optimization of a 7-dof baxter robot: global design sensitivity and step size analyses". *The International Journal of Advanced Manufacturing Technology,* ***93***(9-12), December, pp. 4153–4167.

[30] Bagheri, M., Naseradinmousavi, P., and Morsi, R., 2017. "Experimental and novel analytical trajectory optimization of a 7-dof baxter robot: Global design sensitivity and step size analyses". In ASME 2017 Dynamic Systems and Control Conference, American Society of Mechanical Engineers, American Society of Mechanical Engineers, p. V001T30A001.

[31] Bagheri, M., Krstić, M., and Naseradinmousavi, P., 2018. "Joint-space trajectory optimization of a 7-dof baxter using multivariable extremum seeking". In IEEE American Control Conference (ACC 2018), pp. 2176–2181.

[32] Bagheri, M., Krstić, M., and Naseradinmousavi, P., 2018. "Multivariable extremum seeking for joint-space trajectory optimization of a high-degrees-of-freedom robot". *Journal of Dynamic Systems, Measurement, and Control,* ***140***(11), p. 111017.

[33] Serra, J., and Vincent, L., 1992. "An overview of morphological filtering". *Circuits, Systems and Signal Processing,* ***11***(1), pp. 47–108.

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

[35] Kingma, D. P., and Ba, J., 2014. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*.