Voice Controlled Object Grasping Robotic Arm for Visually Impaired Disabled Veterans

Rashika Natharani
Computer Science
California State University, Fullerton
Fullerton, USA
rashika.natharani@csu.fullerton.edu

Francis Liri
Computer Engineering
California State University, Fullerton
Fullerton, USA
lirifx@csu.fullerton.edu

James Samawi
Computer Engineering
California State University, Fullerton
Fullerton, USA
jsamawi@csu.fullerton.edu

Henry Lin
Computer Engineering
California State University, Fullerton
Fullerton, USA
henrylin14@csu.fullerton.edu

Kayla Lee
Computer Engineering
California State University, Fullerton
Fullerton, USA
kaylalee@csu.fullerton.edu

Nate Ruppert
Computer Engineering
California State University, Fullerton
Fullerton, USA
nate.ruppert@csu.fullerton.edu

Kiran George
Computer Engineering
California State University, Fullerton
Fullerton, USA
kgeorge@fullerton.edu

Anand Panangadan
Computer Science
California State University, Fullerton
Fullerton, USA
apanangadan@fullerton.edu

***Abstract*— Robotic arms have been growing in popularity for various applications like for the manufacturing industry, medical industry, and aerospace industry. This paper utilizes a low-cost robotic arm, DOBOT Magician, to help visually impaired individuals access essential items from a refrigerator. The system takes in audio input using natural language, i.e., without the need to learn special vocabulary from a user and converts it so the system can identify the object using a depth sensing camera and a robotic arm that retrieves the item based on the audio input. This system can properly identify and retrieve the correct item using audio input through natural language processing, a modified version of the deep-learning object detection network YOLOv4 (You only look Once) and a custom-made Python library for the DOBOT Magician robotic arm using Inverse Kinematics to position the arm to the position of the item to retrieve.**

Keywords—natural language processing, speech-to-text, object detection, assistive robotics, automation, robot kinematics

I. INTRODUCTION

Robotic arms have been used in many different scenarios, each serving a specific purpose. Some examples include arms that assist with manufacturing cars, welding, opening doors or helping the physically impaired. To accommodate these specific jobs, the robotic arms vary in size from a few inches to tens of feet like the Canadarm on the International Space Station. More recently, robotic arms are being used in the biomedical industry in different assistive applications [1].

In particular, assistive robot systems are being developed to enable physically disabled persons to interact with their home environment. It is tedious for physical impaired persons to do everyday tasks like accessing the refrigerator and retrieving the correct item. Although caregivers can aid the physically impaired, there is a growing burden on caregivers because caregivers need to take care of another person in addition to themselves [2]. When the disabled person also has a vision impairment, there is an additional challenge – how to specify the part of the environment that needs to be manipulated? For instance, a visually impaired person may struggle to open the refrigerator and identify one desired item from the many items inside the refrigerator. The most prevalent method for visually impaired individuals to accomplish this daily task is by labeling items with braille paper. Using braille paper or tactile bumps, visually impaired individuals can identify a few objects; however, increasing the number of objects limits its practicability [3]. This method also requires the assistance of another individual to properly identify and assist with the labeling process; however, if a visually impaired individual attempts this task on their own it may lead to mislabeling. Other methods include meticulous memorization techniques, which become increasingly difficult with age. Thus, an assistive robotic system that does not have a steep learning curve is desired to assist visually impaired persons with physical disabilities.

We describe the design of an assistive robotic arm-based system to help individuals with impairments that focuses on grabbing items of daily life use and bringing these items to the user. The system relies on analyzing audio input from the user to accurately identify a desired object. The system has a speech to text component from which voice commands can be extracted and translated into action. Those actions, for the purpose of this study, will be focused on tasks that can be completed in a kitchen environment, such as grabbing a water bottle from the refrigerator. The eventual goal is to evaluate the usefulness of the proposed system to blind military veterans but is also applicable for those with other physical impairments. Implementation of the system requires both development of hardware components and then the software components to control them. This includes the voice recognition, object detection, and inverse kinematics components.

Our implementation uses a DOBOT Magician robotic arm and gripper. The DOBOT Magician robotic arm has four joints: the base, the rear arm, the forearm, and the rotation servo. An Intel D435i stereo camera was used to feed real-time 3D images of the surrounding environment to an object detection program, which was trained using the YOLO framework and a dataset of previously taken photos of the bottles. A voice recognition

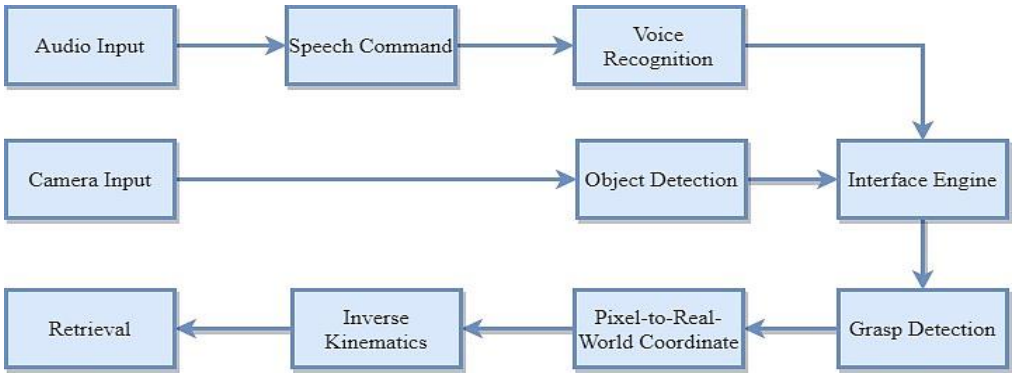

Fig. 1. Proposed model of overall framework

program converts the user's speech to text and identify commands to send to the object detection component.

II. Related Work

Assistive robotic arms in the research scope have been implemented with various methods and user-friendly features. The Wheelchair Mounted Robotic Arm [4] takes a six degrees of freedom robotic arm to go, allowing for portable computer vision assisted object detection and object retrieval for the disabled users. Attempts to remove manual inputs to control a robotic arm have gone as far as implementing a brain-machine interface for user input. The Assistive Robotic Arm Control based on Brain-Machine Interface [5] has turned this daunting task into a reality, allowing users to think of the object's location and the computer vision interface will aid in the robotic arm's retrieval of the object.

III. System Architecture

The architecture of the proposed system is shown in Fig. 1, and includes voice recognition, object detection, and grasp detection components to enable a user to obtain an object from a simple voice command. Operation of the system begins with the user recording a voice command using a microphone, which is converted to a text command in the voice recognition engine. Then, the text command is passed to the interface engine along with the output of the object detection generated from the camera input. The interface engine creates a unique task that combines the information from the voice recognition and the object detection. This information is then used to determine which handle needs to be grasped in the grasp detection. Once that is determined, the pixel-to-real-world coordinate of the handle is found and translated using inverse kinematics to then retrieve the object using the robotic arm gripper.

A. Voice Recognition

The user's commands can be converted from speech to raw text using the speech recognition library; the NLP engine can then be used to convert the raw text into text that can be recognized in the robot's world model. The NLP engine then takes the object's physical description and action from the raw text to be used as input for the interface engine; these are extracted using a sentence segmenter, tokenizer, lemmatization, coarse-grained POS, and semantic mapping. The sentence segmenter first divides the raw text into two separate sentences. The two sentences are then separated into individual words at the tokenizer [6]. The resulting words from the tokenizer then go through a lemmatization process, which takes just the base of the word [7][8]. The result of the lemmatization then goes through the coarse-grained POS process, which labels each token as a noun, verb, or adjective. The last step is to do semantic mapping, which maps each of the tokens in an action-object-description form using the respective POS classifications.

B. Object Detection

Object detection obtains the information from a camera as input and detects the image frame. The engine outputs various information into a JSON format like the object type, color, and pixel coordinates to be used in the interface engine. A modified YOLOv4 is used with a darknet in Python. A modified model is used to reduce the memory consumption to be lightweight and to reduce the time required to for object detection. This would increase the recognition time. Integer computations are used to replace floating point and decrease the total size of the network by 159 MB [9]. The model's result is then placed through the color detection module and the pixel coordinate detection module to extract the color and pixels of the detected objects respectively. The results are saved in a JSON file to be fed to the inference engine.

C. Grasp Detection

Once the object of interest is recognized, Deep Convolutional Neural Network (DCNN) is utilized to predict the suitable grasp location. A graph configuration consists of the location and orientation of the gripper and is represented using five dimensions approach introduced by Lenz et al[10]. A single step prediction technique is used where a DCNN[11] predicts the grasp configuration using the entire RGB image. The Single shot detector technique assumes that there is only one graspable point in the object and therefore process the entire image in one go making it faster. The benefit of using only RGB channels instead of RGB-D is to faster processing and less overhead.

$$g = \{x, y, \theta, h, w\} \quad (1)$$

The center of the rectangle is given by *(x, y),* θ corresponds to the orientation of the rectangle relative to the horizontal axis, *(h, w)* is the height and width of the rectangle. For the robotic arm used in this paper the *h* and *w* remain fixed. The Grasp predictor model is derived from the fifty-layer deep residual model,

ResNet50, which is used to extract features from the 3-channel image. In the ResNet50 model, two fully connected layers with rectified linear unit (ReLU) as activation functions are used in place of the last single connected layer. These fully connected layers act as the baseline model and use features extracted by the ResNet50 to predict the grasp configuration.

D. Object Location with Stereo Camera

Using a trained model, we can perform object detection on 2D image data to identify different potential grasp points on each of the relevant objects detected. The object detection can be performed on streaming data however to perform the grasping action, the robotic arm needs to know the distance to the object. This information is provided in the depth data. We use the Intel RealSense D455 Camera [12] to generate the streaming video for object detection and to provide the depth information. The camera can capture depth rate up to 90fps, but we use 30fps and use the minimum resolution 640x480.

The camera obtains the depth information by using active infra-red stereo technology. The depth information is generated by performing dense stereo matching based on features identified using visible light and infra-red [13]. The output point's x-distance, y-distance, and depth information (D) is provided in meters. However, some challenges with the camera include the fact that the accuracy needs to be improved and sometimes depth information is not provided for the selected point in the frame.

The final information sent to the robotic arm includes the object bounding box width, height, and distance to object. If any additional information is provided by the object detection model (e.g., object orientation) this can also be passed to the robotic arm. Fig. 2 shows the depth, object ID, confidence level, object height and width superimposed on the object bounding box.

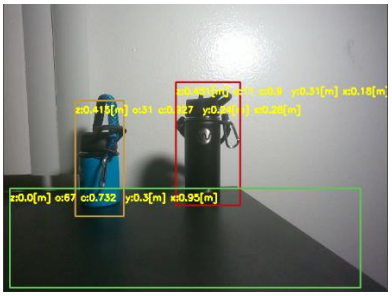

Fig. 2. Object depth and size captured using Intel RealSense Depth Camera D455.

IV. Experiment Setup

The Dobot Magician robotic arm, as shown in Fig. 3, has four joints: base joint, rear arm joint, forearm joint, and end-effector joint. These joints can be seen in Fig. 3 and labeled J_1 to J_4, and the direction of rotation denotes the positive direction. There are two link lengths (a_1 and a_2) that can be found: the first link is between J_2 and J_3, and the second is between J_3 and the bottom fastener (shown between J_3 and J_4).

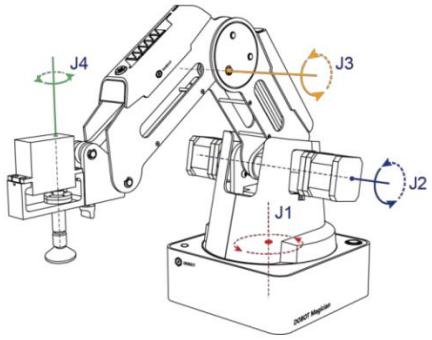

Fig. 3. Dobot Magician robotic arm mathmatical model

To grasp a bottle handle using the Intel D435i stereo camera and the Dobot Magician requires additional computation between the camera and robotic arm reference frames, as well as the robotic arm to the target depth-point location. Rather than using the four-coordinate X, Y, Z, R system, the four joints and joint angles were used to control the arm's movements. This can be accomplished by using the properties of a homogeneous transformation matrix [14] and inverse kinematics [15]. Initially, the x, y, z real world coordinates of the target depth-point with respect to the robotic arm reference frame A will be found using the x, y, D real world coordinates of the camera (reference frame B). Afterward, the 2-D target-depth point with respect to reference frame A with coordinates z, y and the link lengths of the two-link robotic arm will be used to find the angles needed to rotate both the link joints to reach the target-depth point.

A. Training the Voice Recognition Engine

The voice recognition engine uses the user's voice command as an input, and the commands are converted from speech to text using the Google Speech to Text application programming interface (API). The text command is then used as the input to the natural language processing (NLP) engine, which provides a semantic mapping of the input in the form object-action-description. The NLP engine was trained using the NLTK library and the Spacy library in Python using a training dataset of 100 English sentences.

B. Training the Object Detection Engine

The object detection engine is taught to pick out the color and pixel coordinates into a JSON format from an image. This allows for the engine to detect the objects defined in the color and pixel coordinates. A modified YOLOv4 algorithm is used to train the engine with captured images datasets. The original dataset consisted of 331 images and the training is divided into a 70-10-20 split for training, validation, and test dataset. The first dataset are plain images without any filters applied. The following dataset uses labelled images rotated ±45° without any filtering. Rotating the images guarantee that the object detection is functional in cases where the camera or the objects are tilted. Finally, the third case is used to simulate a low light environment where the image brightness is reduced by 30% to determine the effectiveness of the engine.

C. Training the Grasp Detection Engine

To train the grasp detection model, a custom dataset is used to improve the accuracy of prediction. The grasp detection

dataset is different from the object detection dataset as it only has 1 bottle in the frame. The dataset is annotated like the Coronell grasping dataset. Each image has multiple suitable grasp rectangles. There are total 245 images with 206 images in training set and 39 images in testing dataset. In order to ensure that the network performs well on the images in different orientations and positions, five-fold cross validation is used with image-wise split.

D. Transformation Matrix

Pose (denoted by ${}^{A}\xi_{B}$) is the combination of both position and orientation at a reference frame *B* [14]. Since pose is relative, the notation describes the pose at reference frame *B* relative to the reference frame *A*.

$$ {}^{A}\xi_{B} \sim {}^{A}\mathbf{T}_{B} = \begin{bmatrix} {}^{A}\mathbf{R}_{B} & {}^{A}\boldsymbol{t}_{B} \\ 0,0 & 1 \end{bmatrix} \tag{2} $$

$$ {}^{A}\mathbf{R}_{B} = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}, {}^{A}\mathbf{R}_{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_{\varphi=0} \tag{3} $$

Equation (2) defines pose in more detail, which is a 3x3 homogeneous transformation matrix comprising of two key elements. The translational element ${}^{A}\boldsymbol{t}_{B}$ is a vector of components $<{}^{A}x_{B}, {}^{A}y_{B}, {}^{A}z_{B}>$ which describes the position of reference frame *B* with respect to *A*. The second key element describing orientation is the rotation matrix ${}^{A}\boldsymbol{R}_{B}$ (3) which can be a clockwise rotation, counterclockwise rotation, or no rotation at all.

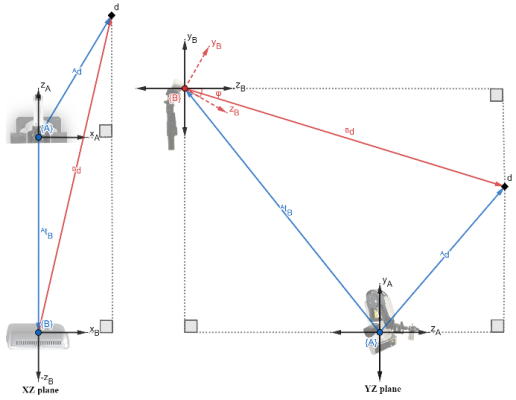

Fig. 4. Translation and rotation of reference frame A and B pointing to a depth-point "d", observed from the XZ and YZ planes.

Fig. 4 shows a 3-D transformation problem that is partitioned to be two 2-D transformation perspectives. The camera is positioned directly behind the robotic arm on a tripod, with a certain translation vector ${}^{A}\boldsymbol{t}_{B}$ and positive pitch angle φ. Reference frames *A* and *B* contain vectors pointing to the depth-point *d* in both 2-D perspectives.

$$ {}^{A}\tilde{\boldsymbol{d}} = {}^{A}\mathbf{T}_{B}\,{}^{B}\tilde{\boldsymbol{d}} \in \mathbb{R}^{2} \tag{4} $$

$$ \begin{bmatrix} {}^{A}x \\ {}^{A}z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & {}^{A}x_{B} \\ 0 & 1 & {}^{A}z_{B} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{B}x \\ {}^{B}z \\ 1 \end{bmatrix} \tag{5} $$

Equation (4) entails the multiplication of the homogenous transformation matrix with the homogenous vector from reference frame *B* is equal to the homogenous vector from reference frame *A* [14]. Specifically, this equation is a subset of the Cartesian product of two real number coordinates in either the XZ or YZ planes measured in meters. In the case of the XZ plane perspective, the equation can further be simplified into its elements and components (5). There is no rotational component, so the 2x2 identity matrix in (3) will be used. The translation vector and homogeneous vector of reference frame *B* are known, so ${}^{A}x$ can easily be found. Since ${}^{A}x_{B}$ is zero (the camera is positioned directly behind the robotic arm) ${}^{A}x$ is equal to ${}^{B}x$ after simplification.

$$ \begin{bmatrix} {}^{A}z \\ {}^{A}y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi & {}^{A}z_{B} \\ -\sin\varphi & \cos\varphi & {}^{A}y_{B} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{B}z \\ {}^{B}y \\ 1 \end{bmatrix} \tag{6} $$

$$ {}^{A}z = \sqrt{D^{2} - {}^{B}y^{2} - {}^{B}x^{2}}\cos(\varphi) + {}^{B}y\sin(\varphi) + {}^{A}z_{B} \tag{7} $$

$$ {}^{A}y = -\sqrt{D^{2} - {}^{B}y^{2} - {}^{B}x^{2}}\sin(\varphi) + {}^{B}y\cos(\varphi) + {}^{A}y_{B} \tag{8} $$

For the case of the YZ plane perspective, (6) is an expansion of (4) using the clockwise rotation matrix for the first case of (3). The positive pitch angle φ, the camera-measured ${}^{B}y$ component, and the translation vector components ${}^{A}y_{B}$ and ${}^{A}z_{B}$ are known. The camera-measured depth *D* coordinate/vector will allow one to find the ${}^{B}z$ component. Pythagoras theorem in 3-D describes *D* as itself squared equaling the summation of the squared vector components $<{}^{B}x, {}^{B}y, {}^{B}z>$. Since all other variables are known, one can make an equation for ${}^{B}z$ easily. Using this knowledge, (6) can be simplified into (7) and (8) to finally find the *y*, *z* coordinates for the robotic arm reference frame *A*.

E. Inverse Kinematics

For all programmable Dobot Magician joints (J_1 ... J_4), there exists a corresponding joint angle (q_0 ... q_3). Joint 1 angle q_0 can be found using the previously calculated *x* and *z* coordinates in reference frame *A*. The angle q_0 is equal to the inverse tangent of the opposite component *x* divided by the adjacent component *z*. The angle q_3 is found from the grasp detection program.

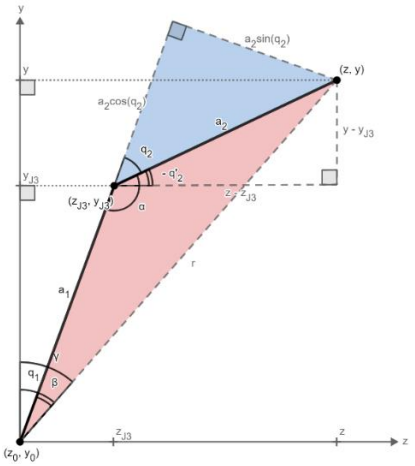

Fig. 5. Inverse kinematics geometrical model

Fig. 5 shows the inverse kinematics mathematical model to geometrically find the angles of q_1 and q_2 using the robotic arm link lengths a_1 and a_2 and the YZ plane target depth-point d (under the assumption z_0 and y_0 are zero) [15].

$$\|{}^{A}\boldsymbol{d}\| = r = \sqrt{(z - z_0)^2 + (y - y_0)^2} \tag{9}$$

$$r^2 = a_1^2 + a_2^2 - 2a_1a_2\cos\alpha \tag{10}$$

The variable r is the radial distance between the origin and the depth-point d (9) which is equal to the magnitude of the vector ${}^{A}\boldsymbol{d}$ in Fig. 5. Since all side lengths of the red triangle in Fig. 5 are known, the Law of Cosines [15] can be used to find the unknown angle α opposite to the side length r (10).

$$\cos\alpha = \frac{a_1^2 + a_2^2 - r^2}{2a_1a_2} \tag{11}$$

$$q_2 = \cos^{-1}\left[\frac{r^2 - a_1^2 - a_2^2}{2a_1a_2}\right] \tag{12}$$

Equation (11) is the rearrangement of (10) in order to substitute α and solve for q_2. Examining Fig. 5, one can find that α is simply π minus q_2. Substituting α in (11) with this new expression, it can further be simplified using a trigonometric identity, causing the cosine function to be negative and only containing the parameter q_2. Finally, q_2 can be isolated (12) by making (11) negative and performing an inverse cosine on both sides.

$$\beta = \tan^{-1}\left[\frac{a_2\sin(q_2)}{a_1 + a_2\cos(q_2)}\right] \tag{13}$$

$$q_1 = \tan^{-1}\left[\frac{z}{y}\right] - \beta \tag{14}$$

To find the last angle q_1, the surrounding angles β and γ must be found first. The angle β can be found by concatenating the blue right-angle triangle seen in Fig. 5 to the red triangle, creating one large right-angle triangle of both red and blue triangles [15]. From the perspective of β, the adjacent length is the addition of both a_1 and $a_2cos(q_2)$, and the opposite length is $a_2sin(q_2)$, meaning a trigonometric relationship can be established as seen in (13). The angle γ can be easily found using another inverse tangent of the opposite length z divided by the adjacent length y. Both γ and β are used to find q_1 which is simply γ minus β (14), implying q_1 is dependent on the angle q_2.

Note that the origin for J_3 lies on the horizontal line segment $z - z_{J3}$ in Fig. 5. For the sake of robotic joint programmability, the negative angle $-q'_2$ can be found by first calculating the lengths z_{J3} and y_{J3} using q_1 and the link length a_1. Subsequently, $-q'_2$ can be found trigonometrically using its right-triangle shown in Fig. 5.

V. Result and Error Analysis

The result and analysis of the performance of the robotic grasp location, voice recognition engine, and object detection engine, and grasp detection engine are described next. Python version 3.7 32-bit was used to call, create, and edit the implemented functions from the DobotStudio application, and wrote scripts detailing the robotic arm's movements.

A. Robotic Grasp Location

Three targets were placed to the in front of the robotic arm and to the left, center and right of it. These three cases were documented to find the reliability of the grasp location of the robotic arm. Table I shows the result of comparing initially calculated locations verses the actual manually measured locations from the robotic arm's origin for all cases.

TABLE I. Comparison between stereo camera output location and observed end effector location

Cases and Errors	Initial & Final Cartesian Measurements from Robotic Arm Reference Frame {A}					
	x_i	x_f	y_i	y_f	z_i	z_f
Left Case (m)	0.180	0.181	0.003	-0.002	0.403	0.391
Left Absolute Error (m)	0.001		0.005		0.012	
Left Percent Error	0.56%		N/A		2.98%	
Center Case (m)	0	0	0.088	0.042	0.409	0.446
Center Absolute Error (m)	0		0.046		0.037	
Center Percent Error	0%		52.27%		9.05%	
Right Case (m)	-0.171	-0.193	0.011	0.030	0.376	0.343
Right Absolute Error (m)	0.022		0.019		0.033	
Right Percent Error	12.87%		N/A		8.78%	

The Dobot Magician robotic arm used in this experiment can be seen in Fig. 6 in more detail with its modified horizontal gripper. The absolute and percent errors tell us the end-effector of the robotic arm has a larger chance of missing the target the further right the location of the target is. It is the combination of inaccurate depth readings from the Intel Realsense D435i stereo camera and sensitive positioning of the camera that produced results that were off by a few centimeters which can be crucial for grasping bottle handles and other objects.

Fig. 6. Dobot Magician with 3-D printed horizontal servo motor bracket for extended reach and horizontal gripping capabilities.

B. Voice Recognition engine:

The accuracy and average time to process voice command using NLTK and Spacy library is compared in Table II below.

The accuracy is considered as one when the action and object are both extracted correctly.

TABLE II. SPACY VS. NLTK PERFORMANCE FOR NLP ENGINE

Parameters	**Spacy**	**NLTK**
Avg time to complete the workflow	20 ms	77 ms
Precision	0.73	0.48
Recall	0.67	0.64
F-Score	0.69	0.62

Spacy package has higher precision, recall, and F-score as compared to NLTK. Also, average time to complete for NLTK is more than Spacy because it takes more time to parse as compared to Spacy package.

C. Vision engine:

The mean average precision (map) for all three datasets: original dataset, rotated dataset, and light adjusted dataset is compared in Table III below. Here it is observed that the average precision is comparable in all three cases, but the average loss is greater for a dataset with augmentations.

TABLE III. COMPARISON OF MAP AND AVG. LOSS OF ALL DATASETS

	map%	**Avg loss**
Plain Dataset	99.8%	0.1730.
Rotated Dataset	99.8%	0.5067
Light Adjusted Dataset	99.6%	0.2577

D. Grasp Detection Program

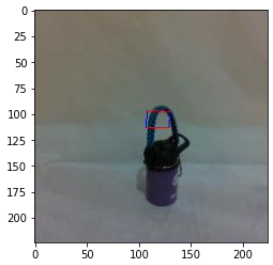

Fig. 7. Object depth and size captured using Intel RealSense Depth Camera D435i.

The rectangle grasp metric evaluates the entire grasp rectangle, and a grasp is considered successful if the difference between the predicted and ground truth grasp angles is less than 30 degrees and the Jaccard correlation coefficient between the predicted and ground truth grasps is greater than 25%. An example of the grasp detection is shown in Fig. 7. In case of grasp detection, while training the Deep Convolutional Neural network average loss for validation set is 24.4469 and for test set is 71.26. The mean average precision calculated using the Jaccard correlation coefficient for the test set is 41.02%.

VI. CONCLUSION AND FUTURE WORK

Robotic grasp location, voice recognition engine, vision engine, and grasp detection program all show very promising results. Further improvements can be made to increase the accuracy of the stereo camera, and the Dobot Magician program to go to the specified depth location with more accuracy. The next step for this research is to combine all aspects of voice recognition, object and grasp detection programs, and robotic arm hardware into one user-friendly system.

REFERENCES

[1] J. Cornejo, J. A. Cornejo-Aguilar and R. Palomares, "Biomedik Surgeon: Surgical Robotic System for Training and Simulation by Medical Students in Peru," 2019 International Conference on Control of Dynamical and Aerospace Systems (XPOTRON), 2019, pp. 1-4, doi: 10.1109/XPOTRON.2019.8705717.

[2] S. Gushi, Y. Shimabukuro and H. Higa, "A Self-Feeding Assistive Robotic Arm for People with Physical Disabilities of the Extremities," 2020 5th International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), 2020, pp. 61-64, doi: 10.1109/ICIIBMS50712.2020.9336390.

[3] Afb.org. 2021. Guidelines for Prescription Labeling | American Foundation for the Blind. [online] Available at: <https://www.afb.org/blindness-and-low-vision/your-rights/rx-label-enable-campaign/guidelines-prescription-labeling> [Accessed 21 April 2021].

[4] P. Karuppiah, H. Metalia and K. George, Automation of a Wheelchair Mounted Robotic Arm using Computer Vision Interface," IEEE Instrumentation and Measurement Society (I2MTC), 2018.

[5] K. Shim, J. Jeong, B. Kwon, B. Lee and S. Lee, "Assistive Robotic Arm Control based on Brain-Machine Interface with Vision Guidance using Convolution Neural Network," IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6-9 October 2019.

[6] S. Jang, H. Shin, E. Park, S. Choi and C. Jeong, "A ContextAware Citation Recommendation Model with BERT and Graph Convolutional Networks," 15 Mar 2019.

[7] S. Ahangama, G. T. Weerasuriya and M. Nandathilaka, "A Rule-based Lemmatizing Approach for Sinhala Language," in 3rd International Conference on Information Technology Research (ICITR), Moratuwa, Sri Lanka, 2018.

[8] A. K. Ingason, S. Helgadóttir, H. Loftsson and E. Rögnvaldsson, "A Mixed Method Lemmatization Algorithm Using a Hierarchy of Linguistic Identities (HOLI)," Advances in Natural Language Processing. GoTAL, vol. vol 5221, 2008.

[9] Y. J. Wai, Z. b. M. Yussof, S. I. b. Salim and L. K. Chuan, "Fixed Point Implementation of Tiny-Yolo-v2 using OpenCL on FPGA," International Journal of Advanced Computer Science and Applications, Vols. vol. 9, no. 10, 2018.

[10] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," The International Journal of Robotics Research, vol. 34, no. 4- 5, pp. 705–724, 2015.

[11] S. Kumra, C. Kanan , "Robotic Grasp Detection using Deep Convolutional Neural Networks," Computing Research Repository, 2017, vol. 1611.0.

[12] RealSense, I. (2021, March 12). Introducing the Intel® REALSENSE™ depth CAMERA D455. Retrieved April 22, 2021, from https://www.intelrealsense.com/depth-camera-d455/.

[13] RealSense, I. (n.d.). Intel RealSense D400 series product Family Datasheet. Retrieved April 22, 2021, from https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet?_ga=2.131134799.1339093438.1619074329-1868482062.1619074329.

[14] P. Corke, "Describing rotation and translation in 2D," Robot Academy. Retrieved April 22, 2021 from https://robotacademy.net.au/lesson/describing-rotation-and-translation-in-2d/.

[15] P. Corke, "Inverse Kinematics for a 2-Joint Robot Arm Using Geometry," Robot Academy. Retrieved April 22, 2021 from https://robotacademy.net.au/lesson/inverse-kinematics-for-a-2-joint-robot-arm-using-geometry/.