Pet-caring Robot

Jun-Min, Wu
Department of Computer Science, National Taiwan University
Taipei, Taiwan
r10922132@ntu.edu.tw

Ho-Chi, Hsieh
Department of Computer Science, National Taiwan University
Taipei, Taiwan
r10922110@ntu.edu.tw

Yuan-Xiang, Deng
Department of Computer Science, National Taiwan University
Taipei, Taiwan
r10922105@ntu.edu.tw

Ho-Ling, Chang
Department of Computer Science, National Taiwan University
Taipei, Taiwan
r10944055@ntu.edu.tw

***Abstract*—In this paper, we propose a robot system intend to be used in pet-caring application. We combine our robot system with AWS cloud service which can control robot remotely. We devise Lex chatbot as user interface which user can give an order with it. After sending command to robot, we use object detection technique to detect which one is the task need and control our robot arm to complete the task. As our experiment show, our pet-caring robot can complete each task correctly in a simple manner. Hence, we believe our robot can benefit pet owner greatly.**

Keywords—pet caring, cloud service, robot arm

I. INTRODUCTION

Nowadays, pet play an important role in our daily life. People also think that pet can let their life better and relieve their pressure at workspace. Hence, the demand of taking care of pet got every people concern [1]. However, most of the people don't have enough time to take care of their pet because owner always have to work for a long time or can't go home accidently. But, there are still some daily routine to do such as feeding pet, playing with pet and watering plant. In this situation, we should ask some of our friend to help us do this daily routine. This always take a lot of efforts.

Robotic techniques have been used in many scenario such as factory, self-driving car, health care and so on. One of the characteristic of robotic techniques is that it can execute some motion repeatedly in an intelligent manner. So, we decide to use robotic technique to solve the problem that we can't take care of pet in person. We propose to service four type of pet which is cat, dog, fish, plant respectively. Also, we provide three types of motion in according with each pet.

A. Watering

This function is mainly to water plant in case withered. First, the robot arm should grad the watering device and move it to plant position. Next, we will adjust the angle of water device to let water pour into potted plants. Last, we will place it to the original position.

B. Feeding

This function is to pour pet food to the plate. Considering that pet may grab pet food automatically, we decide to put a cover on the pet food jar. Hence, we devise another motion to take the cover to another place and put the cover back to the jar.

C. Playing

As everyone know, some pet like to play with ball. Our robot also provided this function. First, the robot arm grad a plate which there is a ball on it and swipe the plate quickly to let the ball threw away.

Moreover, cloud service will be integrated into our system. we use AWS as our cloud service because it provides some extra functions which is useful for our system. We use Lex chatbot to communicate with robot arm. However, we also devise reservation function which can reserve a task in advance. Also, after completing task, we can ask AWS to show the record photo to ensure everything done well.

II. RELATED WORK

A. Automatic Speech Recognition

Automatic Speech Recognition (ASR) is an important research field, which helps generating transcription from speech data. Previously, most people used HMM-based methods which require low computation resource and still remain good performance. In recent year, people are more interested in deep learning-based ASR models. One example is Baidu's Deep Speech model [2] using CNN plus RNN-based architecture with CTC loss algorithm that can demarcate each character of the words in speech. Another example is RNN-based sequence-to-sequence model which divides spectrogram of audio data into pieces and considers each piece as one element in the sequence e.g. Google's Listen Attend Spell (LAS) model. In this study, we use an attention-based model called Lex provided by Amazon Web Services (AWS) for ASR.

B. Slot Filling

Slot filling refers to the process of completing information in order to transform user's intention into several clear instructions and it is a sequence labeling task. After slot filling, the input word sequence is tagged with the slot labels. There are several strategies for slot filling, such as 1-of-N encoding and beyond 1-of-N encoding. In addition to this kind of intuitive methods, we can use neural network (NN) for slot filling as well. Approaches for slot filling include CNN [3], RNN [4], BLSTM [5] and so on. This study applies Lex from AWS, which supports slot filling with a chatbot interface.

C. Cloud Service

Since the computation requirement is growing rapidly due to development in artificial intelligence (AI), cloud services are getting popular nowadays. Moreover, for intelligent robot, it is expected to perform some complicated instructions and also can be remotely controlled. As a result, more studies are trying to apply cloud services into robot so that it can break through the limitation of local computation resource and take advantages of other services over internet [6]. Besides, some studies perform robot-to-robot cooperation and create a complementary environment for the robots using cloud services [7].

D. Object Detection

Computer vision is a key component of robotics that is indispensable for most tasks performed by robot. For object detection in computer vision, there are many types of methods to such a task, like the well-known Fast R-CNN [8], Mask R-CNN [9], and YOLO [10]. The methods mentioned above are taking advantages of CNN, which is widely used for computer vision tasks. In this study, we do not utilize this kind of CNN-based models for detecting and localizing the target objects since our work just needs to recognize few specific objects, like dog dish, flowerpot, and fish tank.

III. METHODOLOGY

The overview of our proposed system is shown in Fig.1. In this section, we will specify the details of each component in the system.

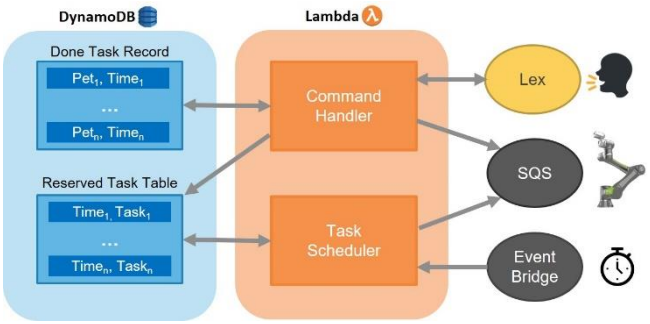

Fig. 1. The overview of our proposed system.

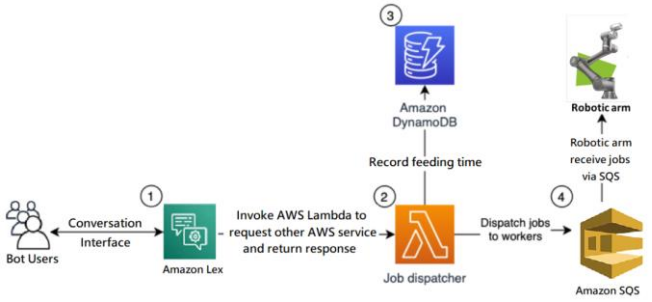

Fig. 2. The architecture of timely jobs. Amazon Lex gets bot users' intent and invoke an AWS Lambda function. An Amazon DynamoDB table stores the feeding time. The Lambda function updates or queries the table for the feeding time and sends tasks to be performed at that time to Amazon SQS.

Fig. 3. Spoken or typed phrases that invoke the intent command robot arm.

pet	RECORD
cat	[{"S":"2022-01-16"},{"S":"10:00"}]
plant	[{"S":"2022-01-17"},{"S":"16:41"}]
dog	[{"S":"2022-01-17"},{"S":"16:38"}]
fish	[{"S":"2022-01-17"},{"S":"16:44"}]

Fig. 4. Table in DynamoDB for recording feeding time.

A. Cloud Service

This section presents our architecture and implementation according to different tasks: timely task, scheduled task and viewing image.

1) Timely Task: The architecture of the timely task is shown in The Fig. 2. The following Amazon services are used:

a) Amazon Lex: Powered by the same technology as Alexa, Amazon Lex provides users with the tools to tackle challenging deep learning problems, such as speech recognition and language understanding, through an easy-to-use fully managed service. In our bot, we design four intents: command robot arm, record feeding time, ask feeding time, and request photos. Amazon Lex understands the user needs according to utterance, a part of which that invoke the intent command robot arm are shown in Fig. 3. The slot {task} can be feed, water, or play; the slot {pet} can be cat, dog, fish or plant; The slot {now} is not required. After getting the required slots, Amazon will trigger AWS Lambda.

b) AWS Lambda: AWS Lambda is a serverless, event-driven compute service that lets users run code without provisioning or managing servers. Amazon Lex integrates with AWS Lambda which you can use to easily trigger functions for execution of your back-end business logic for data retrieval and updates. Here we take an intent of feeding cat now as an example, lambda function will update the feeding time with the current time, where the UTC time is converted to UTC+8, in Amazon DynamoDB, a cloud storage service provided by Amazon is introduced in detail in next section. If the slot now is not filled, the lambda function will then ask the users the reservation time and then store the scheduled jobs in another DynamoDB. The subsequent step and implementation detail is introduced in section B. Scheduled task.

c) Amazon DynamoDB: To store last feeding or watering time. Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. While the intent of asking feeding time is invoked, Amazon Lex can trigger the Lambda function to get the time from Amazon DynamoDB. The records stored in the table are shown in Fig. 4.

d) Amazon SQS: To send or receive messages between cloud and local. Amazon SQS (Simple Queue Service), a message queuing service, provides queues for high-throughput, system-to-system messaging. After receiving a user intent of command robot arm with slots now, AWS Lambda will send a message containing pet and task to Amazon SQS. And robot arm can receive the message via Amazon SQS and execute the instruction.

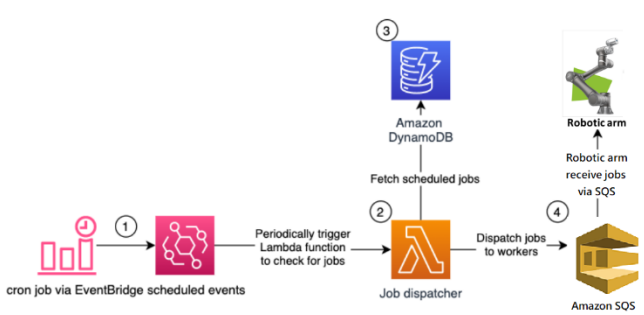

Fig. 5. The architecture of scheduling jobs.

pk	sk	detail
j#2022-01-14T18:50	2022-01-14T18:51:40.123Z	{ "pet" : { "S" : "dog" }, "task" : { "S" : "feed" } }

Fig. 6. Table in DynamoDB for storing scheduled jobs.

2) Scheduled Task: Fig. 5 shows the architecture of the scheduling jobs. Amazon EventBridge with scheduled expressions periodically starts an AWS Lambda function. An Amazon DynamoDB table stores the future jobs. The Lambda function queries the table for due jobs and sends them to Amazon SQS. The following Amazon services are used:

a) Amazon EventBridge: To initiate the serverless scheduling solution. Amazon EventBridge is a serverless event bus that makes it easier to build event-driven applications at scale. It can also schedule events based on time intervals or cron expressions. The fastest rate possible is once every minute.

b) AWS Lambda: To execute the scheduler logic. The Lambda function queries the jobs from DynamoDB and sends them to AWS SQS.

c) Amazon DynamoDB: To store scheduled jobs. The DynamoDB table in this solution has a partition key "pk" and a sort key "sk". For the Lambda function, to be able to query all due jobs quickly and efficiently, jobs must be partitioned. Start with "sk", this is the UTC timestamp for the due date of the job in ISO 8601 format (YYYY-MM-DDTHH:MM:SS). The "pk" looks like the ISO 8601 timestamp in the "sk" reduced to date and time in hours and minutes. The minutes for the partition key must be an integer multiple of five. This value represents the grouping of the jobs, so they can be queried quickly and efficiently by the Lambda function. An example of a scheduled job stored in the table is showed in Fig. 6.

d) Amazon SQS: To send or receive messages between cloud and local. This part is the same as part *A. Timely Task*.

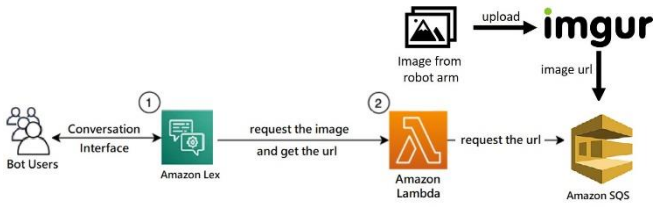

Fig. 7. The architecture of the viewing image.

3) Viewing Image: We provide the function to view photos after the task is completed. The architecture of the viewing image is shown in The Fig. 7. The following Imgur API and Amazon services are used:

a) Imgur: Since Lex can not handle the messages with image files, we utilize Imgur, which can upload images to it and receive url back. In this way, the local robot can just send or receive images through these cloud services easily with a few codes.

b) Amazon Lex: When ChatBot understands users want to view the photo, it will trigger the Lambda function to receive the image URL and response the users with the photo.

c) AWS Lambda: When Lambda function get the instuction to get the image URL, it constantly try to receive the Amazon SQS until receiving a message.

d) Amazon SQS: Here we create a queue dedicated to passing image urls. The message will be sent by local and received by Lambda function in cloud.

B. Image Processing

In this part, we try to use the image captured by the camera on TM-900 to locate centers of each target object. As mentioned before, in our proposed system, the objects that need to be recognize are few and specific. Thus, we do not use NN-based model for object detection and use traditional method to do image processing.

First, we convert the image to grayscale image and set a threshold for binarizing. After binarization, we try to find the contours and use moments to calculate the center of each contour. Since the contour of fish tank may be incomplete due to the light in this way, we use the minimum bounding rectangle (MBR) to reconstruct its shape as shown in Fig. 8.

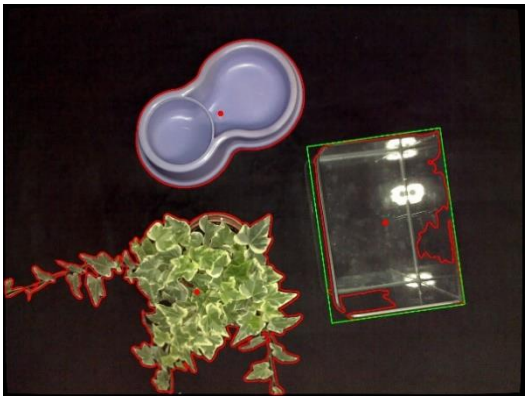

Fig. 8. Result of image processing. Red line draws the original contours; green line is the MBR of found contour of fish tank.

C. Coordinates

There are two different coordinate systems in the TM5-900 robot arm: robot coordinates and camera coordinates. The robot coordinates are responsible for the coordinates of the arm movement in the real world, while the camera coordinates are the position of the object in the picture.

Thus, the position (x_{pixel}, y_{piexl}) obtained from the picture captured by the camera has to be transformed into arm coordinates (x_{robot}, y_{robot}). The following transformation matrix T_{pixel}^{robot} is used for the transformation task.

$$T_{pixel}^{robot} = \begin{bmatrix} -0.732 & 0.711 & 373.733 \\ -0.697 & -0.72 & 817.651 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

D. *Pets-Caring Arm Motion*

The goal of Pets-Caring robot is to achieve the care of all pets in the home. Therefore, the main abilities of the arm are treating different pets with different suitable motions. Up to now, Pets-Caring robot are able to take care of three different kinds of pets: dogs, cats and plants with two different tasks: feed, water and play. The relationship between pets and tasks are showing in Fig. 9.

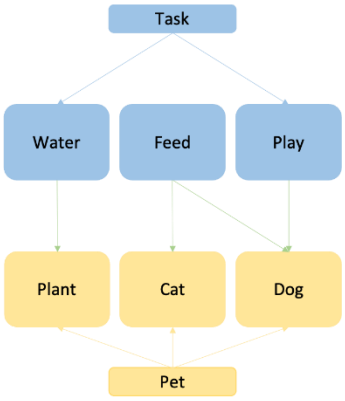

Fig. 9. The relationship between pets and tasks.

Besides, before performing the algorithm we have to record the location of the tool firstly. In other words, the location of flower pourer, fish feed, dog feed, cat feed and ball need to be recorded. After the record is completed, we designed a set of treatment for different tasks to be applied to the corresponding pets. From a mission perspective, the main movements algorithm is shown below:

1. Take a panoramic photo of all the objects (fish tank, flower pot, food container).
2. Calculate the target center point in real time.
3. Use a robot arm to complete tasks.
4. Put the tool back in place.
5. Return the arm and take a picture of the completed task.

In addition to calculating center point and performing arm to the points, there is one more issue that needs to be concerned. That is the offset between object center and tools. Because when the arm is doing the movement like tilting, it will generate a distance from the center, and thus this distance represents an error. To deal with this problem, we measure the offset caused by each task and add this offset to the value of the center point to eliminate the error when the arm is on a mission.

$$(x_{offcor}, y_{offcor}) = (x_{ori} + 80, y_{ori}) \quad (2)$$

$$(x_{offcor}, y_{offcor}) = (x_{ori} + 70, y_{ori} + 10) \quad (3)$$

$$(x_{offcor}, y_{offcor}) = (x_{ori} + 57, y_{ori} - 13) \quad (4)$$

To be more clearly, (2) is applied to task feed with pet dog/cat, (3) is applied to task feed with pet fish and (4) is applied to task water with pet plant. While the rest of the actions keep the original calculated values.

IV. Experiment and Result

The whole system can be divided into two parts. In this section, we will detail the entire work flow of our system. First, we will introduce how to interact with the chatbot built by Lex from AWS and the details of robot arm controlling will be specified in the second part.

A. *Chatbot*

Since our goal is to create a robot that can help people take care of their pet and plant at home, it is important to provide a chatbot for users to take control of this robot. In addition, we want to make it easier and more flexible to communicate with our robot, so the chatbot is expected to be used using both speech and text. With ASR and slot filling, the chatbot can understand which task should be done. Moreover, the chatbot also checks whether the information of the command is enough or not. If some information is missing, the chatbot will ask users to answer the required questions as shown in Fig. 10. Once the task was done, user can ask the chatbot for showing a picture as shown in Fig. 11.

Besides, we can also reserve tasks at certain time as shown in Fig. 12. Since EventBridge will periodically trigger Lambda function, there will be only one or two minutes error between the time when the robot actually completes the task and the scheduled time.

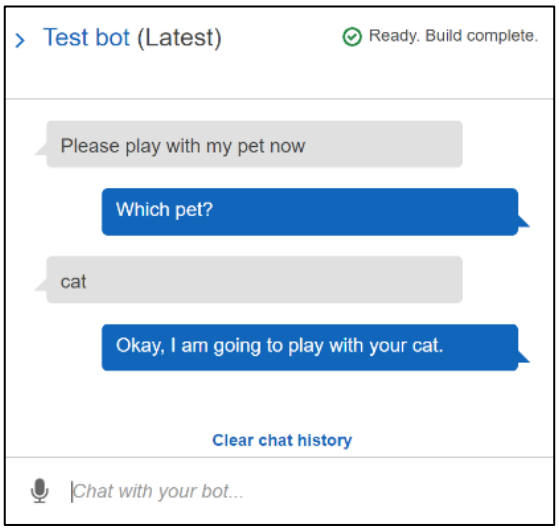

Fig. 10. Example for missing information

Fig. 11. Example for sending a picture after task was done.

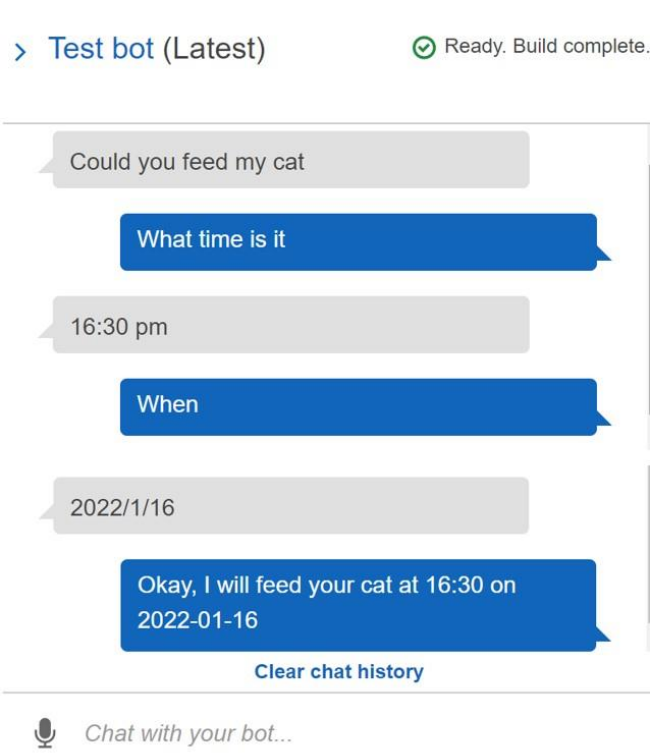

Fig. 12. Example for reserve task

B. *Robotic Arm*

Due to the current limitation of the camera range of the armed camera, we can only place three objects together as shown in Fig. 13. Besides, the tools are arranged in a fixed position as shown in Fig. 14.

As for the arm server, it will constantly listen requests from the SQS queue to check if there are any new instructions to complete. Once the arm receives a new task, it will immediately begin to implement the corresponding tasks, or at the scheduled time to take action. For example, Fig. 15 is showing the task of feeding the dog by arm step by step.

Fig. 13. Full objects view.

Fig. 14. Placement of tools

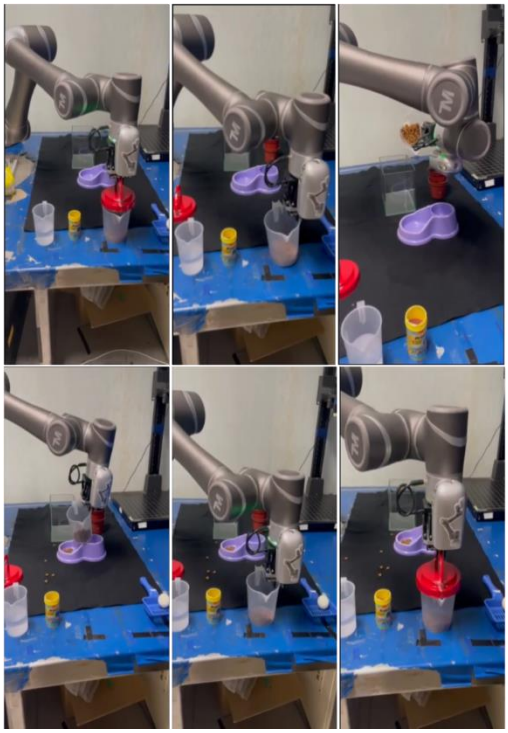

Fig. 15. Process flow design of feeding the dog. The procedure starts from the top left corner and proceeds to the bottom right corner, which means open the cover, grab the feed box, pour into the feed bowl, replace the feed bowl and replace the cover respectively.

V. CONCLUSION

In this project, we have implemented a pet-caring robot integrated with cloud service AWS which let us to control our robot arm remotely. We devise a chatbot Lex to let user give an order in natural language interface which is more friendly. In this chatbot we can make three command which are completing immediate task, making reservation, recording respectively. As for the robot arm motion we implement three functions: watering, feeding, playing. After adjusting hyperparameter precisely, we can complete all these tasks correctly. Our experiment also show that pet-caring robot can be use in real scenario. In future, our work can be separate into two parts. First, we will develop more complicate chatbot which user can give an order with more complicate natural language form. Also, we want to devise more function to let user take more care of their pet. Second, we will let our robot arm do more complicate motion because we only experiment our robot in a short table. To sum up, we firmly believe that our pet-caring robot can benefit the user who keep pet but have problem companying with their pet all the time a lot.

DEMO VIDEO AND SOURCE CODE

A. *Demo Video:* *https://drive.google.com/file/d/1O6KoRFMRoUfrQYESAnPIsR0AnZi-4VnL/view?usp=sharing*

B. *Source Code:* https://drive.google.com/drive/folders/1ZMdmUPuyKO-LXny4Snwm3nwrZKWl4UK6?usp=sharing

WORK DISTRIBUTION

A. *Jun-Min, Wu (R10922132):* Arm Control

B. *Ho-Chi, Hsieh (R10922110):* Arm Control

C. *Yuan-Xiang, Deng (R10922105):*

Cloud programming (Lambda), ChatBot (Lex), database (DynamoDB), message queueing (SQS), and Scheduling (using EventBridge, DynamoDB, and Lambda)

D. *Ho-Ling, Chang (R10944055):*

Image processing, communication between local and cloud, and report.

REFERENCES

[1] Park C.W., Seon JH., Kim JH., Kim JH. 2017 Pet Care Robot for Playing with Canines Robot Intelligence Technology and Applications 4. Advances in Intelligent Systems and Computing, vol 447, ed Kim JH., Karray F., Jo J., Sincak P. and Myung H. (Springer, Cham) p 299-306

[2] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. Y., "Deep Speech: Scaling up end-to-end speech recognition," ArXiv e-prints, 2014.

[3] Ngoc Thang Vu., "Sequential convolutional neural networks for slot filling in spoken language understanding," In Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8- 12, 2016, 3250–3254.

[4] Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., ... & Zweig, G., "Using recurrent neural networks for slot filling in spoken language understanding," IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2014, 23(3), 530-539.

[5] Wang, Y., Shen, Y., and Jin, H., "A bi-model based rnn semantic frame parsing model for intent detection and slot filling", 2018, arXiv preprint arXiv:1812.10235.

[6] Carlos Pillajo, Roberto Hincapie and Edison Pilatasig, "Implementation of a network control system for a Robotic Manipulator as cloud service," 2015.

[7] Afrin, M., Jin, J., Rahman, A., Rahman, A., Wan, J., and Hossain, E., "Resource Allocation and Service Provisioning in Multi-Agent Cloud Robotics: A Comprehensive Survey.," IEEE Communications Surveys& Tutorials, 2021.

[8] GIRSHICK, Ross, "Fast r-cnn," In: Proceedings of the IEEE international conference on computer vision. 2015. p. 1440-1448.

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

[10] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You only look once: Unified, real-time object detection," In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016 (pp. 779-788)