Improved Deep Neural Network Object Tracking System for Applications in Home Robotics

Berat A. Erol, Abhijit Majumdar, Jonathan Lwowski, Patrick Benavidez, Paul Rad and Mo Jamshidi

Abstract Robotic navigation in GPS-denied environments requires case specific approaches for controlling a mobile robot to any desired destinations. In general, a nominal path is created in an environment described by a set of distinct objects, in other words such obstacles and landmarks. Intelligent voice assistants or digital assistance devices are increasing their importance in today's smart home. Especially, by the help of fast-growing Internet of Things (IoT) applications. These devices are amassing an ever-growing list of features such as controlling states of connected smart devices, recording tasks, and responding to queries. Assistive robots are the perfect complement to smart voice assistants for providing physical manipulation. A request made by a person can be assigned to the assistive robot by the voice assistant. In this chapter, a new approach for autonomous navigation is presented using pattern recognition and machine learning techniques such as Convolutional Neural Networks to identify markers or objects from images and videos. Computational

B. A. Erol (✉) · A. Majumdar · J. Lwowski · P. Benavidez (✉) · M. Jamshidi
Autonomous Control Engineering (ACE) Laboratories, Department of Electrical and Computer Engineering, The University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA
e-mail: Berat.Erol@utsa.edu

P. Benavidez
e-mail: Patrick.Benavidez@utsa.edu; p_b_2003@hotmail.com

A. Majumdar
e-mail: abhijit.g.majumdar@gmail.com

J. Lwowski
e-mail: Jonathan.Lwowski@gmail.com

M. Jamshidi
e-mail: Mo.Jamshidi@utsa.edu

B. A. Erol · P. Benavidez · P. Rad
Open Cloud Institute, The University of Texas at San Antonio, San Antonio, TX 78249, USA
e-mail: Peyman.Najafirad@utsa.edu

P. Rad
Department of Information Systems & Cyber Security, The University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA

W. Pedrycz and S.-M. Chen (eds.), *Computational Intelligence for Pattern Recognition*, Studies in Computational Intelligence 777,
https://doi.org/10.1007/978-3-319-89629-8_14

intelligence techniques are implemented along with Robot Operating System and object positioning to navigate towards these objects and markers by using RGB-depth camera. Multiple potential matching objects detected by the robot with deep neural network object detectors will be displayed on a screen installed on the assistive robot to improve and evaluate Human-Robot Interaction (HRI).

Keywords Neural network · Computational intelligence · Simultaneous localization and mapping (SLAM) · Multi object tracking · Deep convolutional neural network (DCNN) · Depth camera · Autonomous navigation · Human robot interaction · Human computer interface · Machine learning · GPS denied environment · Real time implementation

1 Introduction

Improvements on computational intelligence in parallel with assistive robotics, and reinforced applications for object identification engines based on visual sensory readings from RGB-D cameras have increased the accuracy of cooperative task assignments in robotics. Moreover, by implementing pattern recognition fundamentals for object classification, object detection, and computer vision, their impacts on human robot interactions are becoming more crucial than ever. A visual representation of the various fields that pattern recognition has played a major contribution along with their established applications can be seen in Fig. 1. Pattern recognition has especially contributed to applications of intelligent voice assistants and Internet of Things devices, such as Amazon's Echo platforms and Google Home products. This has led to today's smart home applications to decrease the difficult necessity of requiring high computational power, and still can handle the task scheduling requirements easily. These devices are amassing an ever-growing list of features such as controlling states of connected smart devices, playing music, managing alarms, recording tasks, and responding to queries. Potential industrial applications of smart digital assistants are numerous; however, applications are limited due to the digital-only nature of the device. On the other hand, more intelligent assistive robots with higher computational power can loosen this constraint and even make another contribution by providing physical manipulation. A request made by a person can be assigned to the assistive robot by the digital assistant; then, the robot performs its duties while keeping the person updated by verbal feedback. Most of the time the problem lies in this stage and raises a question. Did the robot understand the request correctly and do what the user asked them to do in the right way?

Human-Computer Interaction (HCI) focuses on finding answers for how to efficiently design computers with the latest technology that will provide better user interfaces to make users feel more comfortable with them. As its vision states, HCI interests not only human users and their benefits from a system of computers, but also it investigates the way to use integrated hardware and software platforms. The

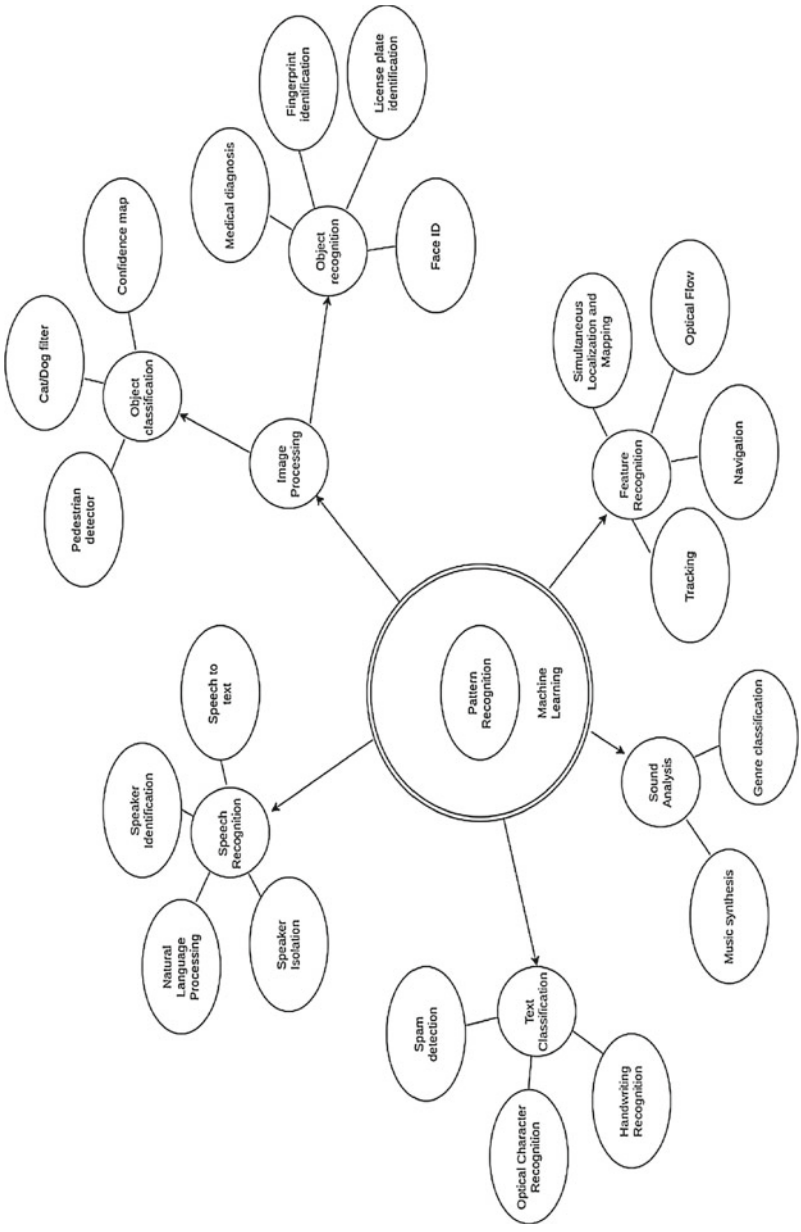

Fig. 1 Application breadth of pattern recognition

goal is to build an interactive relationship between human and machines that are controlled and observed by computers.

Humanoid robots are a well-studied class of robots with the greatest potential in the future to assist with activities of advanced work tasks, and even in a household setting. They are built to perform tasks in a manner that closely follows human form and functionality. Humanoid robots can observe, interact with, and mimic humans. Many have capability to recognize voices, faces, geometric shapes, objects, tools, and environments. They are equipped with very capable sensors, controllers and processing power. Two of the main roadblocks to incorporating humanoid robots are the complexity of robot dynamics and high costs of system components. Hybrids of humanoids and unmanned ground vehicle robots are a popular solution to reduce complexity and the cost of such robotic system. Human-robot interaction (HRI) has increased its importance over the last decades due to the desired conclusion of industrial laws that required higher efficiency and increased productivity. Collaborative working of humans and robots has led to the development of state-of-the-art applications and interfaces for multiple purposes, i.e. robotic arms for the automobile manufacturing lines, unmanned ground vehicles, artificial body parts, and robotic platforms for surgical operations. Since recent developments in the field make robotic systems much more reliable and resilient to changes in the environment, the use of pattern recognition, computational intelligence as a sensory feedback mechanism for a HRI system has become more essential.

Intelligent voice assistants are the central hubs of smart home technologies and are able to perform a variety of tasks. These devices are becoming a must-have for any smart environment applications by featuring list of functionalities, such as controlling the smart thermostats, following a scheduled works, managing smart home observation systems or alarms, and most importantly responding to verbal user queries. What they cannot do is control objects that are not “smart devices”. Home-based assistive robots are the perfect candidates for applications requiring physical manipulation. Steep barriers of cost, limited functionality, and relatively slow performance are preventing the adoption of robots in the home.

In this chapter, we present applications of a proposed intelligent object detection and tracking system for improving functionality and performance of home-based robotic systems. The proposed system is applied to two unique problems: real-time sensing for manipulation and improved environmental awareness for mapping and localization.

Real-time sensing and control is an important milestone on the path towards home based robotic systems. The development of Neural Networks in the past decade has led to substantial increases in the performance of such intelligent networks. Neural networks are a group of nodes that are designed to represent neurons in the brain. These nodes are interconnected and simulate the learning process in the brain. Neural networks are a type of supervised machine learning, which requires the neural network to be trained using labeled data [1]. Slightly more advanced than neural networks, convolutional neural networks are more efficient for classifying images [2]. Convolutional neural networks work by using convolutions with various kernels to detect different features such as horizontal and vertical edges. The convolutional

layers are then stacked on top of each other to detect more complicated features. The final layers of a convolutional neural network are a normal neural network. This is used to combine the features detected by the convolutional layers together to make a classification decision.

Using Deep Convolutional Neural Networks for images, on the other hand, has provided us with trained systems to be able to detect objects of interest with a very high level of accuracy [2–4]. With faster, real-time tracking algorithms [5], the robot can have a better understanding of the dynamics of its environment, and act more responsively, which is desirable in a human-in-the-loop scenario [6, 7]. The benefit of deep neural networks compared to traditional methods to perform these tasks is their ability to adapt better to the object of interest and reject unwanted noise. This makes it ideal for such systems to be incorporated into different environments (e.g. home, office, factory, etc.) without the need to modify the algorithm to adapt to enable detection. Experimental results for tracking objects for manipulation are detailed in the following sections.

Simultaneous Localization and Mapping (SLAM) is a method in robotics to map and navigate GPS-denied environments, such as a home. GPS-denied localization can be a computational constraint for any autonomous navigating task in an unknown environment. Such environments create a problem for locating objects and performing automation operations while creating and following its map by sensory readings, vision sources and etc. This problem requires a system to understand its environment, identify the objects, and localize them as stated in [8]. Sensor data is mapped to odometry data to determine the correct placement in a map. Once a map is developed, a robot can navigate the map and remember where it has gone. Visual SLAM (VSLAM) is the natural extension with visual inputs from one or more cameras. Image based feature extraction is the main method used in acquiring sensory data in VSLAM. In the extraction process, features such as edges and corners from any object are recorded. In cases of complex, dynamic environments where VSLAM will likely be used, landmark selection becomes a difficult problem given a low level of a priori information on the acquired features. Generated maps will contain transient features from objects that moved or disappeared from the environment. With new methods of image classification, namely convolutional neural networks, landmarks can be selected from the environment based off their known properties. The object detector and tracker presented in this chapter is used to find appropriate landmarks for navigation. Preliminary results for the landmark selection process will be presented in this chapter.

In this chapter, an improved object tracking algorithm is proposed for a home-based HRI system. The home-based HRI system presented in this chapter has capabilities of voice and object recognition and Internet of things compatibilities. Exchanges in the system are those associated with the management of tasks in activities of daily living. The chapter is formatted as follows. Section 2 provides a background on the related work. Section 3 details the proposed HRI smart home system. Section 4 describes the prototype of the proposed system. Section 5 details the experimental results with the system components. Finally, Sect. 6 presents the conclusions.

2 Literature and Related Works

A multi object tracking problem mainly consists of two parts: observation model and tracking [9]. The object identification and tracking algorithm used in this chapter uses a convolutional neural network for the model observation part, and uses that data with established tracking algorithms with modifications to suite to our application. Recent developments in object detection using neural networks offers real time performance, which is essential to the application in hand. Networks like You Only Look Once (YOLO) [10], Single Shot MultiBox Detector (SSD) [11], Faster R-CNN [12], R-FCN [13], OverFeat [14], such real-time performance optimal for application in robotics. However, research indicates that the accuracy of such detections reduces with the increase in detection speed [15].

Once the object to be tracked is determined, it is extracted from the frame, thus enabling a smaller region of pixels to be fed into the tracking algorithms to identify the object in future frames. Such tracking, which use feature matching, color segmentation, edge detection, background subtraction etc. can be performed using algorithms like Kanade-Lucas-Tomasi Feature Tracker (KLT) [16], Extended Lucas-Kanade Tracking [17], Online-boosting Tracking [18], Spatio-Temporal Context Learning [19], Locality Sensitive Histograms [20], TLD: Tracking—Learning—Detection [21], CMT: Clustering of Static-Adaptive Correspondences for Deformable Object Tracking [22], Kernelized Correlation Filters [23]. The performance of these methods can be compared and evaluated through benchmarking tools [24] to figure out which one is optimal for one's application. There are several open source libraries which integrate several of these algorithms to facilitate their use in an application like OpenCV [25] and Modular Tracking Framework [26].

Previous related work by the authors of this chapter follows. A framework for navigation and target tracking system for mobile robot was presented using 3D depth image data and used color image recognition, depth camera data and fuzzy logic to control and navigate the robot [27]. Design of a testbed for Large-Scale autonomous system of vehicles was proposed for localization, navigation and control of multiple networked robotic platforms by using cloud computing in [28]. A real-time cloud-based VSLAM was provided in [29] with enhancements to reduce processing time and storage requirements for a mobile robot. A visual SLAM based cooperative mapping study with cloud back-end proposed the importance of the object identification for the mobile navigation and localization [8]. Design and development of a multi-agent home-based assistive robotic system for the elderly and disabled was provided in [30–32]. Furthermore, a cloud architecture for large scale systems of autonomous vehicles was presented in [33]. A foundation for deep neural network control was provided in [34]. An initial deep vision landmark framework was developed for robot navigation by Puthussery et al. in [35]. This system utilized the Inception V3 engine to classify image frames into trained object classes. The most probable detected class was recorded along with positional information relative to the robot in a map. After mapping, objects selected for further inspection were approached by the robot. In this chapter, we utilize a deep neural network based object detector, which has the

capability to detect multiple classes per frame with bounding boxes identifying the detected objects. The use of a real-time object detector greatly improves map resolution, classification throughput, and data acquisition time in the mapping process.

Along with the multi-object tracking algorithms, this chapter also uses unsupervised learning approaches. These unsupervised learning approaches are used for various functions such as object ownership association. The traditional unsupervised learning approaches include k-means [36] and fuzzy c-means clustering [37]. However, in recent years these clustering algorithms have been improved due to some of the traditional methods drawbacks. One such drawback is that both k-means and fuzzy c-means need to know the number of clusters beforehand. In many situations, the number of clusters are unknown. Many different methods have since been developed to remove this constraint. For example, Ester et al. developed Density-based spatial clustering of applications with noise (DBSCAN), which groups together closely packed points [38]. One of the largest advantages of DBSCAN is that it does not require the a priori knowledge of the number of clusters. Another issue associated with k-means and fuzzy c-means clustering is that for large numbers of points, the runtime can be very slow. The typical implementation of k-means has a complexity of $O(\mathrm{N(D+K)})$, where N is the number of points, D is the number of dimensions, and K is the number of centroids [39]. Fuzzy c-means is even slower than k-means, with the typical time complexity of $\mathrm{O(NK^2D)}$ [40]. Although these algorithms can run slow with large numbers of points, some advances have been made to improve this. For example, Kolen and Hutcheson were able to reduce the time complexity of fuzzy c-means down to O(NKD) by removing the need to store a large matrix during iterations, which is significantly faster [41]. Arthur et al. reduced the time complexity of k-means down to $\mathrm{O}(log\mathrm{K})$ by initializing the cluster centers by using points in the dataset that are further away from each other in a probabilistic manner [39, 42].

3 Proposed System

The proposed system is comprised of components which implement the following process. An elderly user makes a request to the voice assistant for a retrieval type task to be completed. In this case, the item to be retrieved is a drink. The task is broken down into its components: action(s), location(s) and object(s). In this example, a robot is tasked to inspect an object. On its way to interact with it, various objects are detected, tracked and mapped. Once candidates for the selected object are detected, a catalog is created for the user to verify. The user provides input to the robot, or voice assistant with a camera, via facial expressions to express satisfaction with the actions of the assistant. Emotion levels are used to select the closest match to the desired output of the system. The robot then completes the task utilizing its physical manipulation capabilities.

3.1 Vision-Based Object Detection and Mapping

Identification and tracking of unique objects requires the following steps. First the object should be detected and classified. Once detected, the object should be tracked frame by frame to ensure that duplicate results are not recorded. Estimates of the objects position are recorded into a map using estimates of the robot pose and properties of the camera. Further detail on each step is provided below.

3.1.1 Multi-object Detection

We use a generalized Convolutional Neural Network (CNN) in our system to perform multi-object detection using an RGB frame captured by a camera on the robot. With the recent development and availability of powerful mobile computers with multi-processing capabilities like the CUDA-cores, we are able to process these frames in real time speeds, to detect multiple objects in a single forward pass of the network. This enables us to use CNN for real time applications like SLAM. The CNN architecture is inspired from open-sourced projects [10–14], the initial layers of which are pre-trained as object classifiers using available datasets of common objects [43–45]. The latter layers of such networks are trained to maximize the Intersection-Over-Union (IOU) of the most likely objects detected in the frame with the bounding box of these objects, also available as supplement to the datasets. We extend these algorithms by adding a higher level of abstracted computer intelligence. For the networks, we use the pre-trained models which are available for most of the networks

3.1.2 Object Ownership Clustering

Sometimes the user may present the robot with ambiguity such as the task of getting the user "their" glasses. This can be an ambiguous task because "their" is a pronoun meaning that the glasses belong to them. "Their" does not provide the robot with any physical description of the glasses, which could cause a problem if multiple people wear glasses in the household. Therefore, ownership of objects could be a very important attribute to consider. For example, if the robot were to see two pairs of glasses, initially the ownership of each pair is unknown so the robot will have to ask the user which glasses is "theirs". Once the robot can determine the ownership of each pair of glasses, the identifying physical descriptors of the glasses can be saved into a database. Since it is very likely that the glasses are placed next to other objects that belong to the owner of the glasses, the robot can assume that the nearby objects also have a possibility of belonging to the glasses owner. To allow the robot to make these assumptions, clustering algorithms can be used.

Once the ownership of an item is verified by the robot, the robot can utilize a clustering algorithm. This algorithm will cluster objects together based on Euclidean

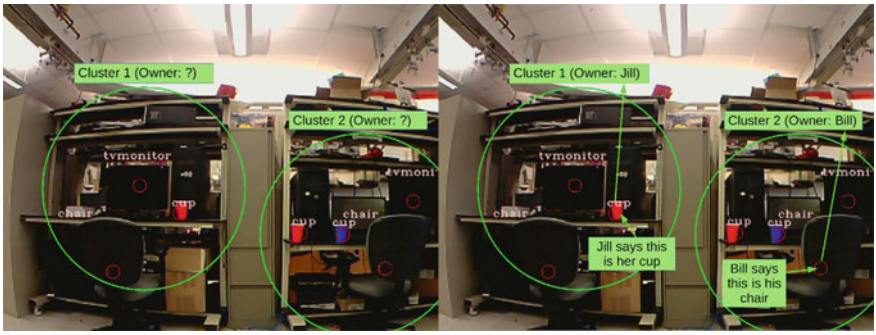

Fig. 2 Example scenario of how clustering can be used to solve ambiguity

distance. Any objects that are in the same cluster as the verified objects can be stored into a database as objects that possibly belong to the owner of the verified object. Now if the user provides the robot with another ambiguous request, the robot can use the objects that were in the same cluster as the verified object to solve the ambiguity. In the same scenario as before, also seen in Fig. 2, if Jill asks the robot to get "her" cup, and there are multiple pairs of glasses, the robot will not know which cup belong to Jill. The robot will then ask Jill to verify which cup is hers. Once Jill responds, the robot will then proceed to get that cup. As the robot is getting the cup, the robot will cluster the objects near the cup. Since Jill's chair and monitor are near her cup, the robot will cluster them into the same cluster as the cup. The robot will then store the ownership of the chair and monitor as having a high probably of belonging to Jill. Now the robot brings Jill's cup back to Jill, and then Jill requests the robot to get her chair. Normally this would be another ambiguous request, but since the robot now knows that the chair was near the cup, the chair has a high probability of being Jill's. This allows the robot to be able to immediately go and get the chair and bring it back to Jill. Since there is still a small chance that the chair is not Jill's chair, the robot will still verify with Jill to ensure that it is actually her chair. A flowchart of the algorithm that the robot can use to solve ownership ambiguity can be seen in Fig. 3.

3.1.3 Object Tracking

Though neural network based multi object detector performs very well as an object tracking neural network, it fails to distinguish between multiple instances of the same type of object that it is detecting. To identify objects for the purpose of automation, we need to track an object. This means to be able to distinguish between two similar objects that might be detected by the multi object detector.

The camera frame will be processed by the multi object detector to provide the location and classification of objects in the frame, while smaller Region-of Interest (ROI) will be selected within the frame based on the same location information to detected features on the object, hence assigning uniqueness to the object. When

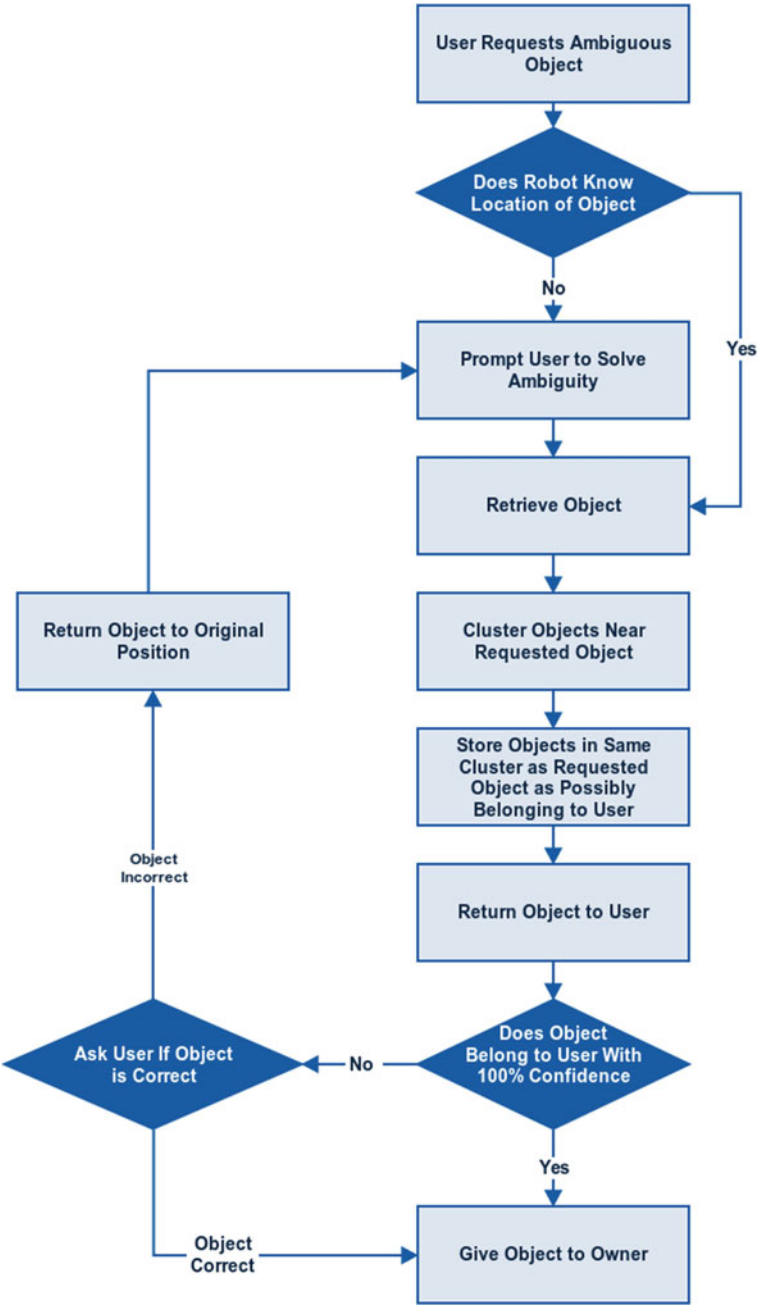

Fig. 3 Flowchart robot will use to solve the ambiguity of requested objects

needed, feature matching is used to solve the problem of ambiguities when two or more objects of the same class overlap, or one object goes out of the camera field of view.

For the purpose of experiments, the number of reliable features to track may be set arbitrarily, and refined to a more experimentally tested decision in the test iterations. A feature management algorithm is utilized to decide on the actions to take in case of loss of features mapped and the minimum number of features required to have a reliable tracking of the object. It is observed that while detecting features of an object, the count of the features may not suffice the need for reliably distinguishing similar objects. To overcome this issue, the algorithm guides the robot to move towards a particular object, once detected, until it has enough features in its feature map. These features of the object are recorded into the memory of the robot along with a picture of the object for reference. A library of all the similar objects and their associated features are stored and then queried with the user to find out which one is of interest. Once a selection is made by the user, other features may be discarded, while the features of the selected unambiguous object is used to track the robot back to the object.

3.1.4 Object Mapping

Positioning of the robot with respect to the environment is important since we need the robot to find its position back to the user and the detected objects. Similarly localizing the detected objects with respect to the environment is also important to plan a path for the robot to maneuver to the object. The kinematic model for the proposed robot's locomotion is a combination of a differential drive kinematics and serial manipulator kinematics. To simplify system development for this current research, the humanoid torso robot was assumed to remain in a fixed pose.

Mapping of the robot and the environment is performed using a combination of different sensors on the robot. Every sensor has different kind of error associated with them. For example, the odometer on the robot is prone to error due to slippage. To overcome this problem, we use an Extended Kalman Filter (EKF) algorithm to perform a sensor fusion between the positions obtained from the odometer, Inertial Measurement Units (IMU), the visual odometry reported by the camera using Simultaneous Localization and Mapping (SLAM) techniques. The use of Robot Operating System (ROS) packages enables us to perform such sensor fusion with minimal effort.

Traditional SLAM algorithms using feature detection are complemented with using multi object detection as reference points on the map to localize the robot. The objects detected by the system act as landmarks in the mapping process. Another EKF is applied to this system to provide a filtered map and localization of the robot. Mapping of the objects is performed once the objects of interest are detected and features are selected and stored. The locations of the objects are stored alongside the features and the picture of the object. With the use of modified neural network based object tracking, we could hand pick a certain category of objects that may offer more

remarkable features to distinguish between other features. Since these features are conglomerated into objects detected by the algorithm, hence the matching process is less intensive than traditional SLAM, and at the same time provide more error correction from similar features, which is a large problem in the field of SLAM.

An important part of integrating an HRI into a system is for the robot to learn. A key way to do this for a task assigned robot is to remember the choice made by the user. As an implementation example, the first time the robot is asked to locate a bottle, and the user selects a specific bottle from a list of bottles the robot found, the specific bottle is stored along with its features, location and a picture for reference in the robot memory. Later, if the user instructs the robot to find the same object, the first guess that the robot makes is of the stores bottle in its memory. The user may want a different bottle and deny the robot, however there is a higher possibility that the user may want the same object again, which in turn improves the confidence of the robot with the user, and hence may make the robot more reliable. This is also useful in saving time for the robot to look for an object that it had already looked for earlier.

3.1.5 Object Database Creation

This section defines the creation of a virtual database of the objects detected, tracked and mapped by the robot. As described earlier, various real time multi-object detector algorithms may be used to track the current position of an object in the frame of the image frame, as observed by the robot. Once the presence of N object is confirmed, each of them are compared to the existing database to find if any of these objects are already in the library. In order to check if the object being recognized is already in the library, it takes into consideration various attributes about the detected object which includes color, features, ownership information, location and last access of the same object. Such attributes about each recognized object is compared with the corresponding attributes of all objects in the database, to compute a confidence level as shown in Fig. 4. This confidence level is then used to determine among three possible actions to be taken:

- If the object is already present in the database, affirm the presence of the same object in the image frame and determine if other action need to be performed on the object, for example pick up the object.
- If the confidence of the same object being present in the database is above a certain threshold but not high enough, compare to the most probable object in the database and update the object attributes in the database with the observed attributes.
- In case the confidence of the object being among the ones in the database is lower than a set threshold, add the object and its corresponding attributes as a new object in the database.

It should be noted that the attributes are weighted, when evaluating the confidence levels. This is because, certain attributes may be more reliable than others. For example, if the robot detects a bottle in its image frame, being an object that can be moved

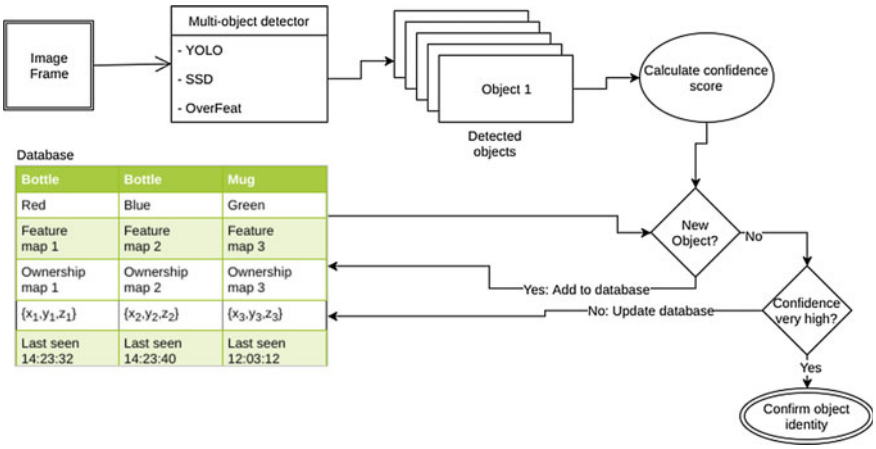

Fig. 4 Object database query, creation and update

since the last observation of the object, we assign a low weight on the location of the object while determining the confidence of the object being the same bottle in the database.

On the other hand, the features recorded for the bottle is assigned a higher weight, since a higher score on feature matching is more accurate indication of the same object being view by the robot image frame. Since we already define the objects that the object tracker can recognize, we also pre-define the weights associated with the different attributes of the corresponding object.

We can hence formulate, Eq. 1, a weighted average confidence level calculation of a detected object as:

$$\begin{aligned} C_n &= \vec{W}\vec{A} \\ \vec{W} &= \left\{w_c, w_f, w_o, w_l, w_t\right\} \text{ and } \vec{A} = \left\{s_c, s_f, s_o, s_l, s_t\right\}^T \end{aligned} \tag{1}$$

where, $C_n \rightarrow$ Confidence level of object n,

$w_c, w_f, w_o, w_l, w_t \rightarrow$ Pre-defined weight vector for different attributes: color, feature matching, ownership, location and last access time, respectfully,

$s_c, s_f, s_o, s_l, s_t \rightarrow$ the attribute scores for color matching, feature matching, ownership, location and last access time.

3.1.6 Determining Optimal Action Sequence

One important decision the robot needs to make, is the sequence of the actions to be taken when multiple commands are requested by the user. This can be difficult because there are a lot of variables that can be considered. To simplify this process, we assume that the robot can only retrieve one object at a time, some of the objects

locations are known, some of the objects locations are unknown, and the user has placed a higher priority on some objects versus others. To determine the optimal action sequence, the robot will first receive multiple requests from the user. Using machine learning algorithms, the robot will to decide whether each request is a low, medium or high priority request. The robot will sort the actions with known locations based on the cost calculated using Eq. 2, where C is the cost of the action, D_{RO} is the distance from the robot to the object, D_{OU} is the distance from the object to the user, and P is the predicted user's priority for that action.

$$C = \frac{D_{RO} + D_{Ou}}{P}$$
$$P_{low} < P_{medium} < P_{high} \tag{2}$$

If there are not any objects with known locations, the robot will search until it finds an object. The robot will then proceed to retrieve the first item, while simultaneously searching for objects with unknown locations. If the robot finds an unknown object, the robot will rerun the auction algorithm and determine the new optimal action sequence. If the robot does not find any unknown objects, then the robot will grab the object and return it to the user while still searching for unknown objects. The robot will repeat this process until all the actions have been completed. A flowchart describing this algorithm in a high level can be seen in Fig. 5.

3.1.7 Avoiding an Obstacle in the Environment

Once the robot has decided the optimal sequence of actions to take, the robot needs to successfully travel to the object to retrieve them. While traveling to the objects, the robot may encounter obstacles. These obstacles need to be avoided in order to ensure the safety of the robot and to not cause damage to the household. To avoid these obstacles, the already onboard camera can be utilized. Using the vision based obstacle avoidance algorithms such as the reactive vision only sliding mode controller developed by [46]. The robot can use its front facing camera to avoid obstacles, while still moving towards the object of interest. The results of the reactive vision based obstacle avoidance algorithm can be seen in Fig. 6.

4 Prototyping Robotic Smart Home System

4.1 Robotic System Hardware

The assistive robot used in this system is a hybrid of an unmanned ground vehicle and a humanoid robot, Fig. 7. For the humanoid portion of the hybrid machine, the humanoid robot torso is used which is a 3D printed open source robot from the

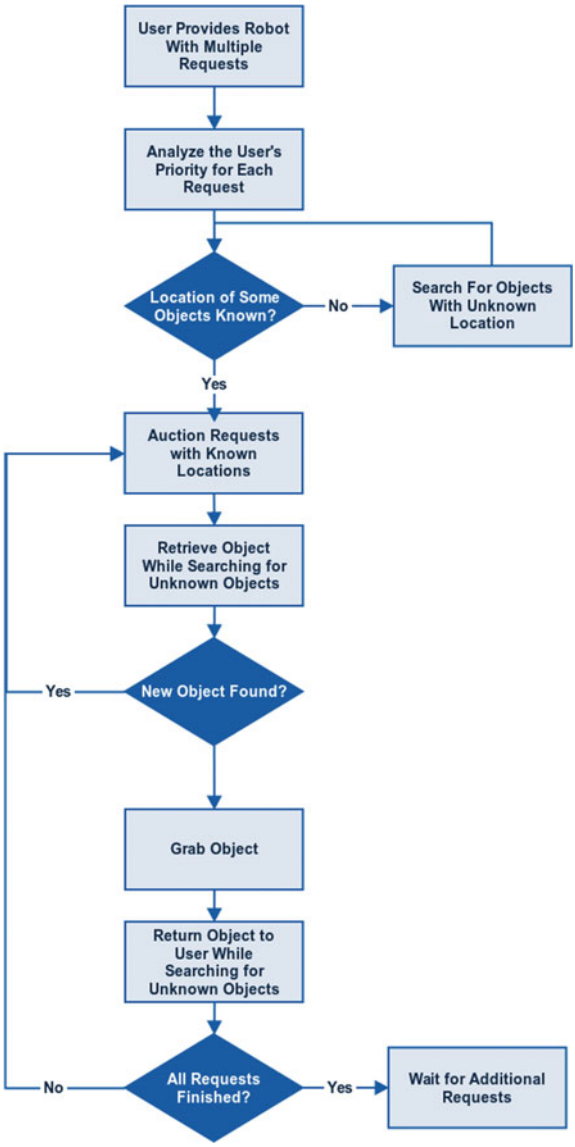

Fig. 5 Flowchart robot will use when the user requests multiple actions

torso up. The humanoid torso was combined with a Kobuki Turtlebot 2 research platform from YujinRobot later on as summarized in Fig. 8, which is an unmanned ground vehicle. The Turtlebot2 was selected due to its customizable capability and open source software. The rover is equipped with a Yujin Robot Kobuki base, a 14.8 V Lithium-Ion battery, and a Hardkernel ODROID XU4 minicomputer. The ODROID XU4 minicomputer was selected as the embedded computer for the UGVs. It features a Samsung Exynos 5422 octa-core CPU, 2 GB DDR3 RAM, USB 2.0/3.0

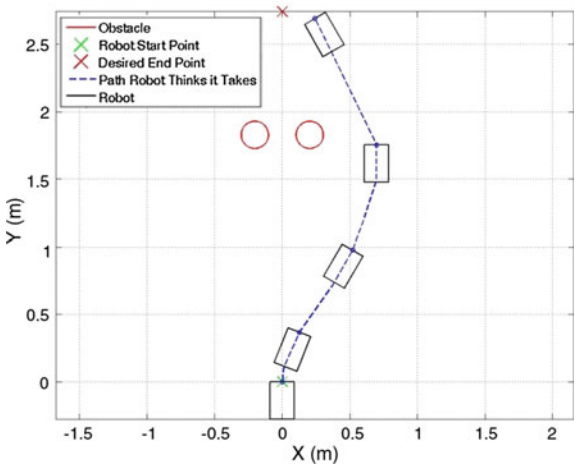

Fig. 6 Results of the vision based obstacle avoidance algorithm developed by Lwowski et al.

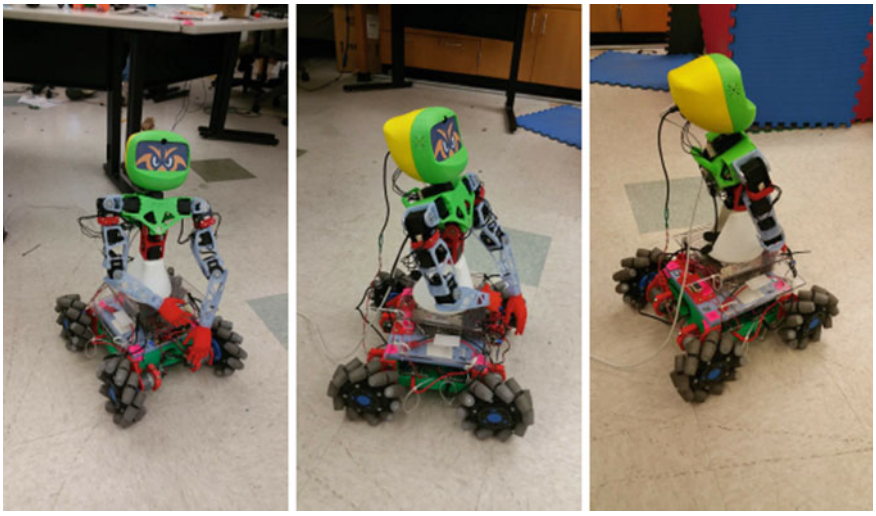

Fig. 7 Preliminary low-cost prototype for hybrid 3D printed mobile assistive robotic system

and a 64 GB eMMC card for storage. A Meanwell DC-DC converter was connected to the Kobuki's 12 V 5A output to supply power for the ODROID XU4 (5 V/4A requirement).

In addition, the rovers come with cliff sensors (left, center, right), wheel drop sensors (left, right), a single axis gyro and motor overload protection. The hybrid robotic platform is compatible with the Robot Operating System (ROS) by extensions of APIs supplied for both research platforms. To obtain better directional awareness, a BOSCH BNO055 Inertial Measuring Unit was added to provide absolute orientation

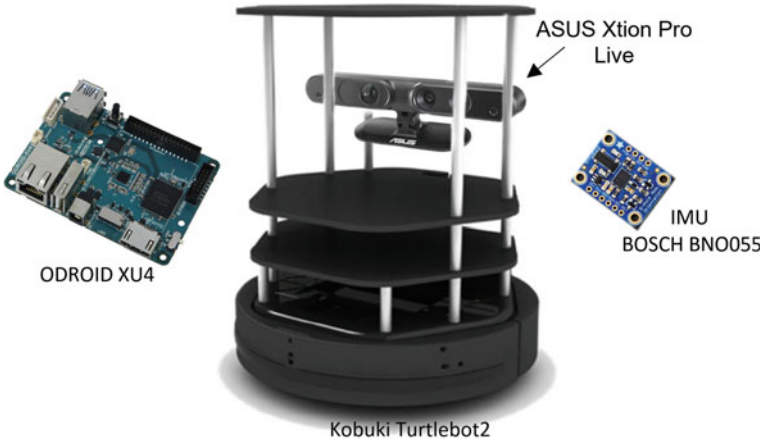

Fig. 8 Kobuki Turtlebot 2 has been chosen and modified with ASUS Xtion Pro Live RGB-D camera, powered with ODROID XU4microcontroller and BOSCH BND055 IMU installed

to the system. This IMU integrates multiple sensors to obtain a stable absolute output: a triaxial 14-bit accelerometer, a triaxial 16-bit gyroscope and triaxial magnetometer.

The humanoid robot torso has been mounted on the Kobuki Turtlebot 2 robot as shown in Fig. 9. A camera mounted in the head of humanoid robot torso is used to detect objects in the environment. Control of the hybrid robot is performed using ROS. The *Kobuki_ROS* package handles control of the base robot. A custom designed head unit was designed to support addition of a five-inch touchscreen LCD display (ODROID-VU5), monocular camera, stereo speakers for synthesized auditory feedback, and a microphone for obtaining commands from the user.

An overview of the prototype system is provided below in Fig. 10. The smart home system includes interfaces for voice, vision, cloud-based computation, and robotic platforms. In this section, preparation of robotic hardware, the HRI, object detection and tracking algorithms and the control loop are discussed.

4.1.1 Human-Robot Interface

Visual Interface

A software interface was developed for the new humanoid robot torso's head unit using pyqt3 for the graphics and ROS for the interface to the data. A ROS software package ace poppy hri display was developed for this work. Inputs to this package are the desired target, the robot state, and text to display. A question and answer game, used in HMI, is implemented by the robot and user of the system. An example of the HRI is displayed on the humanoid robot torso head unit display in Fig. 11. The example shows the user request "check the plant" to the robot, a response "is

Fig. 9 Redesigned prototype of the system with the mobile platform and the torso robot is installed performing object identification and navigation tasks

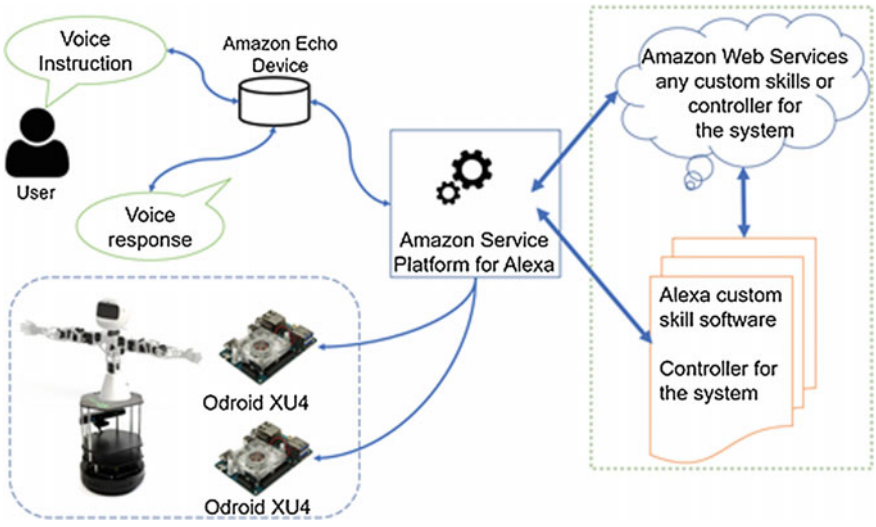

Fig. 10 System representation included simple flow chart for IoT device and voice-activated control

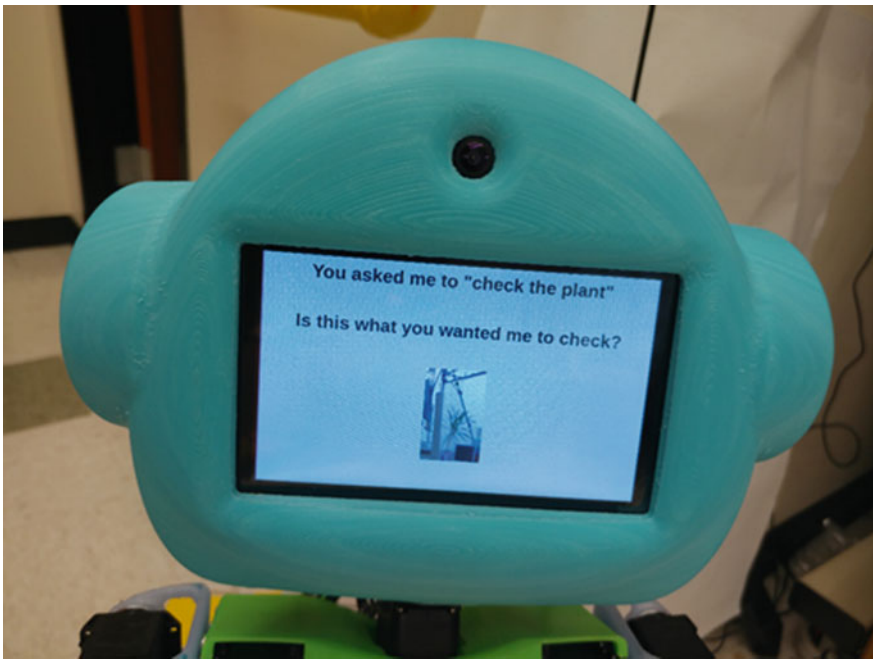

Fig. 11 Question and answer type HRI for a selected cropped output of the database displaying a plant

this what you wanted me to check?", and a picture of the item that the robot checked. Stylistically, the text of the question and answer HRI is like the voice user interface (VUI) de facto standards used in voice assistants. This was implemented like VUI as voice is used to provide the robot tasks and is shown in Fig. 10.

Auditory Interface

Auditory commands are provided to the robot using a home voice assistant, in this case an Amazon Echo Dot. Auditory responses from the robot are generated using a combination of Linux programs *espeak* for synthesis of words into a WAV file and *aplay* for playing back the WAV file. The option to use a WAV file was selected for a combination of reasons. Most important of all, *espeak* tends to connect to the audio service jack-server slowly or fails intermittently, where *aplay* plays back the audio almost instantaneously and consistently. The second reason is to maintain a history of responses to the user for quality purposes. As a synthesizer, *espeak* tends to be limited in its ability to pronounce certain words and people's names. There is a need to sometimes break a word into its phonetic components to synthesize it correctly.

4.2 Object Detection and Tracking

4.2.1 Modification for Compatibility with Video Source

Darknet, the software package for YOLO, was installed on a high-performance desktop computer with an Intel i5 and a NVIDIA GTX-1080 with 8 GB of memory. The desktop computer is the detection engine for the robot platform. The source code for the neural network was modified to allow connections to the camera feed through Wi-Fi. The Robot Operating System (ROS) was used to provide Wi-Fi interface to the camera feed over a protocol similar to TCP called TCPROS. To be compatible with ROS, use of a branched version of darknet written in C++ was necessary. This software branch contains the modifications necessary to generate a shared library file libdarknet-cpp-shared.so and arapaho, a C++ API to the library. ROS packages ace_arapaho and ace_arapaho_msgs were developed by the authors to use the arapaho API. These packages provide the capability to publish the identified objects with labels, timestamps, and the relevant region of interest bounds of the image. An additional input to the package is the object filter list.

Objects in the list are filtered out from the reported identifications. Outputs of the ace_arapaho ROS node are passed to a feature tracking package, developed by the authors, called ace_object_tracker. Inputs to this package are parameters from the motion of the robot, parameters of the camera, and the image ROIs from the multi object detector. This package develops initial models of the detected objects from the inputs provided to it. These models are used to uniquely identify the incoming data as belonging to a unique object.

4.2.2 Feature-Matching Enhanced Object Detector

It is important to note that even though the output of neural network based multi object detectors resemble tracking, it is just a multi-object detection algorithm which works at a very high throughput. Though it is able to detect the location of an object within the image frame, it does not track objects as individual items. Hence in a frame with more than one instance of the same type of object, for example two different bottles, will be tracked as the object bottle, irrespective of their differences. This ambiguity is problematic in cases where the user wants the system to find a specific object. To resolve this issue, we combine traditional object tracking methods with neural network multi object detectors. The system uses multi object detectors to detect objects in the frame, which are then passed onto feature tracking algorithms. Features are then selected from within the bounding box provided by the object detectors for a particular object. These features are then used to identify a specific object when such ambiguity arise.

4.2.3 Confidence Gradient Tracking

Another common issue when using such multi object detectors is the lack of reliable detection in every frame. Since they process every image individually, isolated from the previous image sequence, it often causes alternating loss and detection of objects in the scene. Another problem while using such algorithms is the detection of false positives. We use a method of confidence gradient tracking to overcome both these issues, to achieve a reliable tracking. Our algorithm uses a complimentary filter to smoothen the detection confidence level of a particular object being tracked. The gradient of this filtered confidence level is monitored by the system. While the robot is moving in a particular direction, if the filtered confidence gradient is positive and the filtered confidence level builds up to a set threshold of confidence level, the existence of the object is confirmed. This confirmation of the object in the scene initiates the feature tracking algorithm, which starts to record features of the object, while the robot is moving towards the object. The algorithm instructs the robot to keep moving towards the object until the number of recorded reliable features for the object matches a pre-defined minimum number. Implementation of the algorithm is explained in more detail in Sect. 5.

5 Experimental Results

5.1 Processing Rate for Object Detection and Tracking

Initial performance tests were executed with a direct USB 2.0 connection from the desktop computer to the camera onboard the robot. In this configuration images are processed at about 27 FPS, which is similar to the camera frame throughput. Further tests utilized images transmitted over TCPROS on a Wi-Fi IEEE 802.11 N connection from the robot to the desktop computer. In this configuration, we were able to process image frames around 5 FPS. The drastic reduction in processing rate is solely due to the transmission of raw image data over Wi-Fi. Compressed image streams will be examined in the future to achieve a higher data rate over Wi-Fi. Selected outputs of this result are displayed in Fig. 12 that show both some correct and incorrect detections. For a couple examples, a cardboard box is covered in white paper is labeled a sink, and a large cabinet is classified as a refrigerator. Items that were detected more reliably (e.g. clocks, bottles, chairs, TV monitors, etc.) were selected as target objects for the experiment.

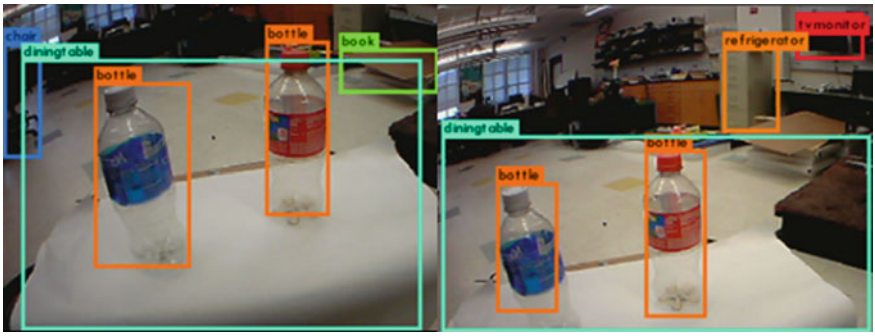

Fig. 12 Selected outputs of YOLO demonstrating correct and incorrect classifications of objects in an image frame

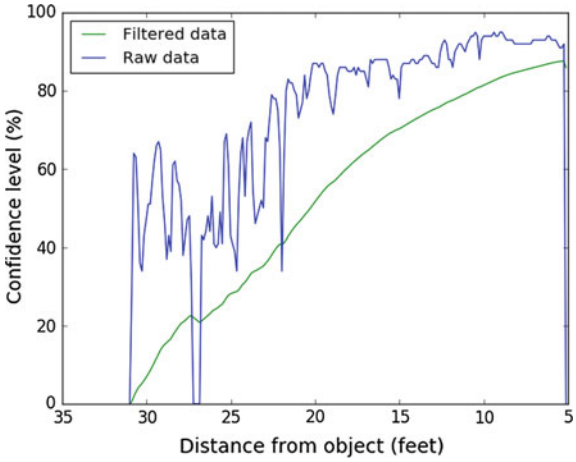

Fig. 13 Detection confidence analysis with respect to distance to object

5.2 Confidence Gradient Tracking

Figure 13 shows a plot of the raw and filtered confidence level of a chair detected, as the camera frame moves closer to the chair. As can be observed from the figure, the raw confidence outputs of a particular object being tracked by the multi object detector is noisy. However, the filtered data shows a general increasing trend of confidence signifying a true positive detection of the object. To recognize this behavior, our algorithm differentiates the filtered confidence level. This differential is used to indicate the increasing or falling nature of the confidence level of a detected object. However, as observed in Fig. 13, there are regions where the algorithm either fails to detect the object or the detection is a false positive.

For this reason, certain regions of the differential are negative. On the other hand, for our algorithm to work, we would want to detect a negative differential slope only

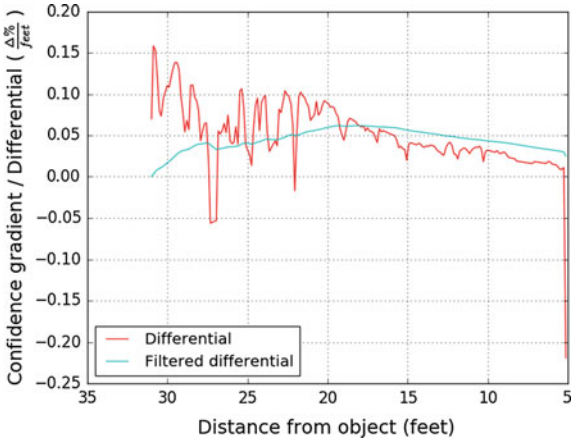

Fig. 14 Filtered differential of the confidence levels with respect to distance to object

when the object for detection is not in the frame. In order to detect a true positive in cases where the object is not detected for few frames, we filter the differential plot as shown in Fig. 14.

If the filtered differential remains positive while the robot is moved towards the object, a true positive of the detected object is established. The filtered confidence data is monitored while the robot is moving towards the object to assign a Region of Interest (ROI) for feature recognition and storage, for matching and identification later. It should be noted that differentiating the original unfiltered confidence levels and then filtering them, generates an output which radically alternates between the positive and negative. This is an expected behavior for a noisy signal.

An example of the differential of the raw confidence levels can be seen in Fig. 15. However, such differential signal cannot be used to distinguish between false positive and a true positive, using the previously explained algorithm. As a result, we use a filtered confidence levels before differentiating the data.

Experimental tests were performed on 3 different objects—person, chair and bottle for three instances each. The algorithm described above was implemented in each case and the results obtained are tabulated in Table 1. Two thresholds were used to determine a true positive. The first one, set to 50% confidence was used to trigger the robot to turn towards the object and move towards it. The next threshold of 60% was used to declare an ROI to start tracking features of the object. The filtered differential was monitored to make sure that the results are not false positives.

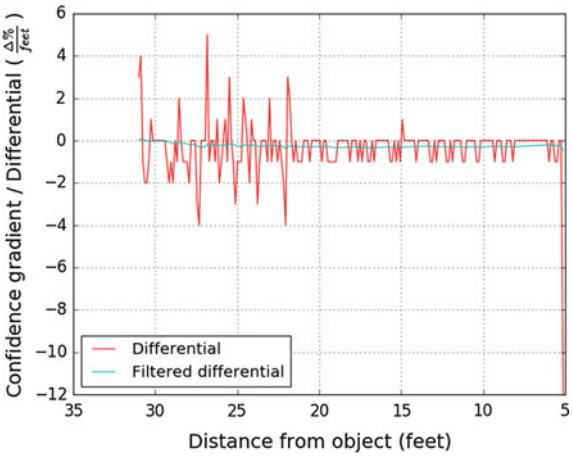

Fig. 15 Differential of unfiltered confidence levels: Alternates rapidly across zero level

Table 1 Confidence gradient tracking (Threshold (true positive/feature recording) = 50%/60%)

Object tracked	Distance from object with true positive affirmation (Feet)	Line of sight angle offset (°)	Maximum features matched for tracking capability	Tracking possible?
Bottle 1	12.77	10	730	Yes
Chair 1	20.26	0	382	Yes
Person 1	20.66	5	127	No
Chair 2	17.18	10	312	Yes
Chair 3	18	0	302	Yes
Bottle 2	5.69	5	N/A	No
Bottle 3	9.55	0	779	Yes
Person 2	20.94	15	911	Yes
Person 3	14.62	10	630	Yes

6 Conclusions

Robotic navigation in GPS-denied environments highly depended on specific approaches for locomotion and navigation tasks. Improvements on computational intelligence tools and pattern recognition approaches, along with reinforced learning applications for object identification engines based on improved RGB-D cameras, have increased the accuracy of cooperative multi-tasking assignments in robotics. Pattern recognition and machine learning techniques, such as Convolutional Neural Networks to identify markers or objects from images and videos, improved the performance in the autonomous navigation and localization experiments.

The use of robotics in the home environment is a very complicated scenario with lots of problems. In this chapter, many problems such as object detection and tracking, object ownership, object mapping, object database creation, determining optimal action sequence, and obstacle avoidance have been addressed. Solving these problems are one of the necessary steps into creating a robust fully functioning home robotic assistant that could be used in our everyday lives. In the future, we plan to integrate all of these decouple systems together, to create a more complete robotic assistant for the home environment. This robotic assistant could then be tested in many different situations in order to gather data and improve our algorithms. These tests would also help us identify new problems that will need to be solved in the future to make the system even more robust and useful.

Acknowledgements The authors would like to acknowledge the support from Air Force Research Laboratory and OSD for sponsoring this research under agreement number FA8750-15-2-0116. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory, OSD, or the U.S. Government. The work partially supported by the Open Cloud Institute at The University of Texas at San Antonio.

References

1. S.B. Kotsiantis, I. Zaharakis, P. Pintelas, Supervised machine learning: A review of classification techniques, 2007, 3–24
2. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105
3. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9
4. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826
5. A. Karpathy, What I learned from competing against a convnet on imagenet, 2014, http://karpathy.github.io/2014/09/02/whati-learned-from-competing-against-a-convnet-on-imagenet
6. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, 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
7. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, in *European Conference on Computer Vision* (Springer, 2016), pp. 21–37
8. B.A. Erol, S. Vaishnav, J.D. Labrado, P. Benavidez, M. Jamshidi, Cloud-based control and vslam through cooperative mapping and localization, in *World Automation Congress (WAC)* (IEEE, 2016), pp. 1–6
9. L. Fan, Z. Wang, B. Cail, C. Tao, Z. Zhang, Y. Wang, S. Li, F. Huang, S. Fu, F. Zhang, A survey on multiple object tracking algorithm, in *2016 IEEE International Conference on Information and Automation (ICIA)* (IEEE, 2016), pp. 1855–1862
10. J. Redmon, A. Farhadi, Yolo9000: better, faster, stronger, 2016, arXiv:1612.08242

11. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S.E. Reed, C. Fu, A.C. Berg, SSD: single shot multibox detector. CoRR, abs/1512.02325 (2015). http://arxiv.org/abs/1512.02325
12. S. Ren, K. He, R.B. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks. CoRR abs/1506.01497 (2015). http://arxiv.org/abs/1506.01497
13. J. Dai, Y. Li, K. He, J. Sun, R-FCN: object detection via region-based fully convolutional networks CoRR. abs/1605.06409 (2016). http://arxiv.org/abs/1605.06409
14. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, Overfeat: Integrated recognition, localization and detection using convolutional networks. CoRR abs/1312.6229 (2013). http://arxiv.org/abs/1312.6229
15. J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, K. Murphy, Speed/accuracy trade-offs for modern convolutional object detectors. CoRR abs/1611.10012 (2016). http://arxiv.org/abs/1611.10012
16. C. Tomasi, T. Kanade, Detection and tracking of point features. Int. J. Comput. Vision (Tech. Rep.) (1991)
17. S. Oron, A. Bar-Hille, S. Avidan, Extended lucas-kanade tracking, in *European Conference on Computer Vision* (Springer, Cham, 2014), pp. 142–156
18. H. Hu, B. Ma, Y. Wu, W. Ma, K. Xie, Kernel regression based online boosting tracking. J. Inf. Sci. Eng. **31**(1), 267–282 (2015)
19. K. Zhang, L. Zhang, Q. Liu, D. Zhang, M.H. Yang, Fast visual tracking via dense spatio-temporal context learning, in *European Conference on Computer Vision* (Springer, Cham, 2014), pp. 127–141
20. S. He, Q. Yang, R.W. Lau, J. Wang, M.H. Yang, Visual tracking via locality sensitive histograms, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2427–2434
21. Z. Kalal, K. Mikolajczyk, J. Matas, Tracking-learning-detection. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1409–1422 (2012). https://doi.org/10.1109/TPAMI.2011.239
22. G. Nebehay, R. Pflugfelder, *Clustering of static-adaptive correspondences for deformable object tracking* (Comput. Vision Pattern Recognit., IEEE, 2015)
23. J.F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters. CoRR abs/1404.7584, 2014. http://arxiv.org/abs/1404.7584
24. Y. Wu, J. Lim, M.-H. Yang, Online object tracking: A benchmark, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013
25. Open source computer vision library, https://github.com/itseez/opencv
26. A. Singh, M. Jagersand, Modular tracking framework: a unified approach to registration based tracking. CoRR. abs/1602.09130 (2016). http://arxiv.org/abs/1602.09130
27. P. Benavidez, M. Jamshidi, Mobile robot navigation and target tracking system," in *2011 6th International Conference on System of Systems Engineering (SoSE)* (IEEE, 2011), pp. 299–304
28. J.D. Labrado, B.A. Erol, J. Ortiz, P. Benavidez, M. Jamshidi, B. Champion, Proposed testbed for the modeling and control of a system of autonomous vehicles, in *2016 11th IEEE System of Systems Engineering Conference (SoSE)*, 2016, pp. 1–6
29. P. Benavidez, M. Muppidi, P. Rad, J.J. Prevost, M. Jamshidi, L. Brown, Cloud-based realtime robotic visual slam, in *2015 9th Annual IEEE International Systems Conference (SysCon)* (IEEE, 2015), pp. 773–777
30. P. Benavidez, *Low-cost Home Multi-robot Rehabilitation System for the Disabled Population* (Google, 2015)
31. P. Benavidez, M. Kumar, S. Agaian, M. Jamshidi, Design of a home multi-robot system for the elderly and disabled, in *2015 10th System of Systems Engineering Conference (SoSE)*, 2015, pp. 392–397
32. P. Benavidez, M. Kumar, B. Erol, M. Jamshidi, S. Agaian, Software interface design for home-based assistive multi-robot system, in *2015 10th System of Systems Engineering Conference (SoSE)* (IEEE, 2015), pp. 404–409
33. S.A. Miratabzadeh, N. Gallardo, N. Gamez, K. Haradi, A. R. Puthussery, P. Rad, M. Jamshidi, Cloud robotics: A software architecture: For heterogeneous large-scale autonomous robots, in *World Automation Congress (WAC)*, (IEEE, 2016), pp. 1–6

34. M. Roopaei, P. Rad, M. Jamshidi, Deep learning control for complex and large scale cloud systems, in *Intelligent Automation & Soft Computing*, 2017, pp. 1–3
35. A.R. Puthussery, K. Haradi, M. Jamshidi, A deep vision landmark framework for robot navigation, in *2017 12th IEEE System of Systems Engineering Conference (SoSE)*, 2017, pp. 1–6
36. S. Lloyd, Least squares quantization in pcm. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
37. J.C. Dunn, A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. J. Cybern. **3**(3), 32–57 (1973)
38. M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (AAAI Press, 1996), pp. 226–231
39. X. Jin, J. Han, *K-Means Clustering* (Springer, US, 2010), pp. 563–564
40. A. Stetco, X.-J. Zeng, J. Keane, Fuzzy c-means++: Fuzzy c-means with effective seeding initialization. Expert Syst. Appl. **42**(21), 7541–7548 (2015)
41. J.F. Kolen, T. Hutcheson, Reducing the time complexity of the fuzzy c-means algorithm. IEEE Trans. Fuzzy Syst. **10**(2), 263–267 (2002)
42. D. Arthur, S. Vassilvitskii, K-means++: The advantages of careful seeding, in *Proceedings of the Eighteenth Annual ACM-SIAM on Discrete Algorithms, SODA*, 2007, pp. 1027–1035
43. T. Lin, M. Maire, S.J. Belongie et al., Microsoft COCO: common objects in context. CoRR abs/1405.0312, 2014. http://arxiv.org/abs/1405.0312
44. M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge. Int. J. Comput. Vision **88**(2), 303–338 (2010)
45. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database, in *CVPR09*, 2009
46. J. Lwowski, L. Sun, R.M. Saavedra, R. Sharma, D. Pack, A reactive bearing angle only obstacle avoidance technique for unmanned ground vehicles. J. Autom. Control Res. **1**, 31–37 (2014). http://dx.doi.org/10.11159/jacr.2014.004