Cognitive Learning Using Distributed Artificial Intelligence

Omkar Pimple, Umesh Saravane, and Neha Gavankar

***Abstract*—We propose a system with multiple mobile agents, which will have a shared intelligence. Such architecture will enable the entire system to become 'smarter' as each individual agent has new experiences and learns about new things. Whenever each node learns something new, it makes its peers learn, thus greatly accelerating the rate of learning of the entire system. The color, shape and size of an image are extracted. An attempt is made to identify the object in the image using its local intelligence. Next, it tries to learn about the object from its peers. If none of its peers know about the object, it simply learns about the object from the user, and updates its own knowledgebase. When a similar object is encountered at a later stage, the system is able to recognize the object based on its own knowledge, or from its peers' knowledge – similar to how humans learn.**

***Index Terms*—Artificial intelligence, recognition, learning, distributed systems, Android, image processing.**

I. Introduction

Cognitive learning in humans is a powerful mechanism that provides the means of knowledge, and goes well beyond simple imitation. It is defined as the acquisition of knowledge and skill by mental or cognitive processes - the procedures we have for manipulating information 'in our heads'. Cognitive processes include creating mental representations of physical objects and events, and other forms of information processing.

Artificial Intelligence (AI) is the area of computer science focusing on creating machines that can engage in behaviors that humans consider intelligent. The ability to create intelligent machines has intrigued humans since ancient times, and today with the advent of the computer and 50 years of research into AI programming techniques, the dream of smart machines is becoming a reality. Researchers are creating systems which can understand speech, beat the best human chess player, and countless other feats never before possible.

An intelligent system simulates a certain form of human reasoning, knowledge, and expertise for a given task, whereas distributed artificial intelligence systems were conceived as a group of intelligent entities, called agents, that interacted by cooperation, by coexistence or by competition. Agents with distinct interests or knowledge can benefit by engaging in negotiation whenever their activities potentially affect each other. Through negotiation, agents make joint decisions, involving allocation of resources, adoption of policies, or any issue of mutual concern. Multiple related issues are typically negotiated at once, with each negotiation issue involving multiple agents [1].

Implementing cognitive learning using distributed artificial intelligence would employ the power of multiple agents in a distributed system to accelerate their learning process rapidly. [2] Such a system would not only be useful in a learning environment, but it could also be used in various applications like networked biometric scanners, security surveillance systems and other similar problem recognition systems.

Manuscript received July 24, 2014; revised October 29, 2014.

The authors are with Department of Computer Science and Engineering, Vidyalankar Institute of Technology, University of Mumbai, India (e-mail: omkarpimple1991@gmail.com, ume7715@gmail.com, nehagavankar1992@gmail.com).

II. Scope of the Project

The scope of the project will be limited to the visual aspects of objects to be recognized and identified. The system will detect and identify objects based on visual attributes such as color, size and shape.

- The distance from the sensor to the object will be approximately the same for each object, in order to avoid ambiguity in the context of the attribute of size.
- The system will be able to detect an object only in a neutral/plain environment. For demonstration purposes, the object will be placed against a plain black background.
- The system will be designed to detect a single object at a time. Support for multiple object detection - if feasible - will be added later on.
- The network connectivity range of the system will be that of a typical Bluetooth network i.e. up to 10 meters. Any agent which is moved beyond that range will be disconnected from the network.

III. Proposed System

The platform for the system will be an Android device. Each agent will be installed on an Android device, which will use the in-built camera as an input sensor, and Bluetooth functionality for setting up a network to interact with other agents in its vicinity. Since an Android device is ubiquitous and portable, it is a perfect candidate for a host which will contain an agent.

As shown in Fig. 1, the project proposes to create a system with multiple agents which are a part of a distributed artificial intelligence. The system will be trained to identify a set of objects based on visual aspects attributes, like color, shape and size. In other words, the system will be given a rudimentary intelligence about a set of objects. The agents will be connected via a wireless network, like Bluetooth. Any agent can leave the network at any point in time, and can return to the network again later.

An object will be detected by the system using its visual sensor i.e. its in-built camera. The agent will try to identify the detected object based on its knowledgebase. If an object

DOI: 10.7763/IJMLC.2015.V5.474

cannot be identified by an agent, the agent will communicate with its neighboring agents and check whether they can identify the object. If none of the agents in the entire system can identify the object, the system will try to learn about the object from the user. It will add the new knowledge about the object in its knowledgebase. Now that it has knowledge about the object, the system has 'experienced' the object, and should be able to identify the object in the future.

If an agent which is outside the network learns about new objects, upon re-entering the network, the knowledge of the entire system will be updated. Thus, even if an agent has never learned about an object, it will still be able to identify it if the object has ever been encountered by any other agent in the system before.

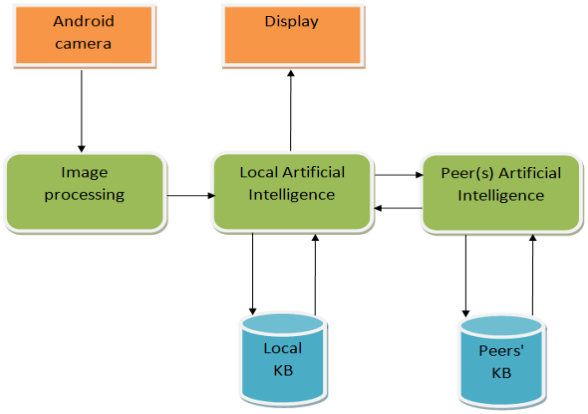

Fig. 1. Proposed system.

The algorithm is as follows:

1) START
2) Acquire the image of the object using the Android camera.
3) Perform image processing and attribute extraction.
   a) Perform isolation of the object from its background.
   b) Extract the most dominant color.
   c) Estimate the size of the object.
   d) Estimate the shape of the object.
4) Attempt to find a match in its own knowledgebase. If a match is found, identify the object and display its name, and STOP. Else, proceed.
5) Get a list of all peers in Bluetooth network.
6) Attempt to learn about the object from an unvisited peer. If knowledge about the object is found, learn about the object, display the object name, and STOP. Else, proceed.
7) Mark the peer visited. Check if all peers have been visited. If yes, go to 7. Else, go to step 5.
8) Learn about the object from the user.
9) STOP

IV. Implementation

The first step in the development of our application was to utilize the Android camera hardware inside the application. There are two ways to do it. One way is to use the built in Android camera application which is provided with all Android phones by default. The second way is to develop a custom camera app, which could be used to use the camera in a programmable manner. Since our application involved the use of a camera only for image acquisition, we did not need a custom camera application. Hence we utilized the built in Android camera for capturing the image.

A. Image Isolation

After capturing an image of the object, the first operation to be performed is object isolation from its background. Isolation in its easiest form would be object segmentation. But segmentation also segments the object itself into multiple sub-segments, which is not desirable. Hence, we decided to work with thresholding the object from its background, provided that the grey levels of the object and the background are in contrast. The isolation algorithm pushes all background pixels to the Hex value 'BLACK', and leaves non-background pixels as they are originally.

The algorithm first analyzes a sample of the background to determine the range of the background grey levels. We implemented this by analyzing the pixels in the first five rows, last five rows, first five columns, and last five columns of the image as background pixels of the image, since these pixels are never going to be a part of the object. The next step would be to set all 'similar' pixels to the lowest grey level - 000000. Doing so keeps the object pixels as they were originally, and blackens the entire background.

Though this method works fairly well in the case of a black background, it performs poorly in case of a light background, due to presence of salt and pepper noise in the background. Hence, to eliminate salt and pepper noise, we implemented a median filter in a separate class, and used it as a part of post processing after performing thresholding. Also, we performed edge detection using Sobel operator, so that we can clearly define the boundaries of the object which will be used for further processing and analysis.

Fig. 2. Image before isolation.

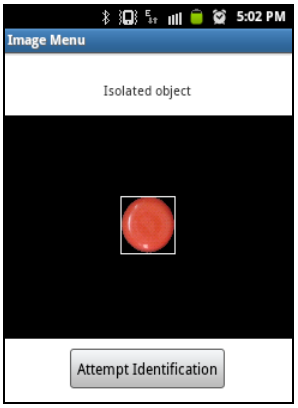

Fig. 3. Image after isolation.

Fig. 2 and Fig. 3 show the screenshots of the application showing the image before and after performing object isolation respectively.

B. Attribute Extraction

The AI uses a part of the image processing module to identify the color, size and shape of the object isolated in the image.

1) Identifying color

A pixel is represented by its red, green and blue (RGB) values - 8 bit each - in a 24 bit integer format. Working with

crisp logic, we would end up with 2^{24} different values for colors, and as many different definitions. Instead, we decided upon a set of frequently used generic colors viz. Red, Orange, Yellow, Green, Blue, Violet, Pink, White and Gray. We used the Hue Saturation Value (HSV) for representing colors. Further, we declared ranges for each of the above colors on the HSV wheel representation. In essence, we fuzzified the entire range of 2^{24} colors to a set of 9 generic colors, based on their hue and saturation.

After defining our set of 9 colors labels, our algorithm analyzed each pixel of the object, and found out which color label each pixel corresponded to by using a distance generator function. The color label in whose range the value of the pixel lied is the color of the pixel. Each pixel of the object is analyzed and the number of pixels corresponding to each color label is calculated.

The next step in the algorithm is to find the most dominant color of the object. This simply translated to finding the color label with the most number of pixels corresponding to it. The most dominant color is the 'Color' attribute of the object.

2) Identifying size

By itself, 'Size' is a highly variable attribute. The size of an object could be large if the perspective is close to the object, or it could be tiny if the perspective is further away from the object. Also, since the system uses only a single camera, there is no way of knowing the absolute dimensions of the object.

Keeping in mind all the above issues, we decided to define 'Size' as the number of pixels the object covers in the image. Further, we decided to fuzzify the attribute into 4 different labels - 'Tiny', 'Small', 'Medium', and 'Large'. The system assumes that the image will be taken from approximately the same distance each time. Taking this assumption into consideration allows us to consider 'Size' as a valid constant attribute of an object.

3) Identifying shape

Identifying the shape of an object was by far the most challenging task to implement. There were no existing libraries for shape recognition in Android. There were a few libraries for real time smile detection, but those were of little use to us in this project. The only other alternative was to develop our own logic for recognizing the shape of the object.

Initially, we tried to detect lines by trying to implement Hough Transform [3]. After line detection, we would look for angles between the lines. If there are 4 angles detected, and each angle is close to 90^{o} , then the shape detected could be a rectangle. There would be other similar rules for recognizing other shape. Not only was Hough Transform difficult to implement, but it would also bring up further challenges like angle detection and vertex detection.

Due to the above challenges, we decided to look for another implementation logic for shape detection [4]. The logic we were developing had to have a good time complexity to ensure that it wouldn't take a long time to execute on the Android platform which has limited processing power and resources.

Our shape recognition logic finally was implemented using the concepts of basic geometry which we had learned back in school. Firstly, we implemented a function which would give us the rectangular borders of the object by considering its topmost, bottommost, leftmost and rightmost limits.

The second step was to calculate the area enclosed by the borders. The third step was to check the relation between the area of the object ('Size') and the area enclosed by the borders. The resulting logic could classify the shape of an object into 3 categories:

- Rectangle
- Circle
- Triangle

The classification 'Rectangle' could be further refined into 'Rectangle' and 'Square' by taking into account the ratio of the height and width of the area enclosed by the border.

One more classification could be defined when the height of the area enclosed by the border is too large as compared to its width, or vice versa. Such a shape could be labeled as 'Oblong'. Finally, when none of the labels can be applied, a default label is used - 'Random'.

Thus, the final labels for the 'Shape' attribute are -

- Rectangle
- Square
- Oblong
- Circle
- Triangle
- Random

Thus, the image processing module isolates the object from its background, extracts the attributes i.e. 'Color', 'Shape', and 'Size'. Fig. 4 shows a screenshot of the application displaying the extracted attributes of the isolated object (which is shown in Fig. 3). The labels are 'Red', 'Small', and 'Circular' for the attributes color, size and shape respectively.

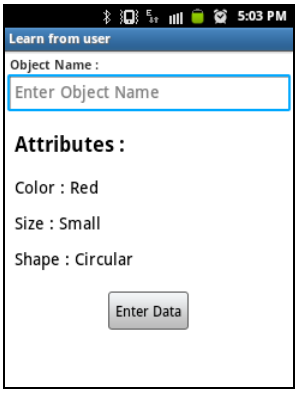

Fig. 4. Extracted Attributes.

C. The Knowledgebase and AI

Android has an inbuilt library for constructing a relational table called SQLite. We used SQLite to create the plinth for our knowledgebase representation.

1) Constructing the knowledgebase

Since we would be referring to the knowledgebase frequently at various places in the application, we created a separate class for the knowledgebase, whose instance could be created as an when needed. As with all other relational database-based systems, an SQLite database needs to be opened before use and closed after use, to prevent loss of integrity. Hence, apart from the usual functions of 'read',

'update' and 'delete', we also had to define open and close functions as public member functions.

In addition to the database related variables, we also created a Database Helper class, which would be used to manage the knowledgebase by ensuring timely closing of the database before the calling activity would be destroyed. Failing to do so would make the application crash due to multiple instances of the same database being open at the same time, which would result in system instability.

2) Using the knowledgebase for identification

The knowledgebase contains information about previously encountered objects by the system. When a similar object is encountered in the future, the AI analyzes the knowledgebase to look for any information about the object at hand. It checks the knowledgebase by first creating an instance of the knowledgebase class, and opening the SQLite based database using which the knowledgebase has been constructed.

The first step it does is use the image processing module to extract the color, size and shape of the current object. After doing so, it looks for information of such an object in its own knowledgebase. If there is an entry where all three attributes match, the system identifies the object. In this case, there is a perfect match between the extracted attributes and the attributes stored in the knowledgebase for the object.

In the event that a perfect match is not found, it looks for 'similar' objects in its knowledgebase which could be potential candidates for identification success. It checks similarity by looking for any entries which have the same shape label, as any entries having different shape labels have a very low probability of being a correct match.

If the shape and color match, but the size does not match, there is a good chance that the object is a faraway or up-close version of an already existing entry. But since such an identification can never be at a hundred percent accuracy, it is always going be a prediction rather than a valid identification. But at the same time, it would not be a good idea to completely discard the possibility of a match, however small that possibility might be. Since there is a chance of this prediction being wrong, the system also offers to make a new entry in the knowledgebase about the object.

In the event that there is no match - complete or partial - with any knowledgebase entry, the system offers to learn about the object from the user. It provides the user with the extracted attributes, a text box to enter the name of the object, and a button to make the system learn about the object by updating the system's knowledgebase with information about the current object, as shown in Fig. 4.

D. Peer-to-Peer Communication

The system engages in peer learning when it encounters an object which it cannot recognize based on the information present in its own knowledgebase. The term 'peer' means another Android device which has the same application installed on it. The peer-to-peer network was implemented using Bluetooth using a custom protocol, as existing protocols such as Mobile Peer to Peer Protocol [5] existed only in simulations.

When a device is unable to recognize an object, it initiates a Bluetooth connection with its peers, and sends a query with the attributes extracted from the current object image. The peer then tries to recognize the object based on the information present in its own knowledgebase. If it is unable to recognize the object, it sends back a negative signal back to the device who initiated the communication.

The source device attempts to query all its peers to find information about the current object. If any of the peer has information about the object, it makes the source device learn about the object i.e. it updates its knowledgebase.

Thus, if even one peer has information about an object which none of its peers know about, the entire system virtually has knowledge about that object, as any peer requiring knowledge about that object can learn from its peers transparently, whenever required.

1) Bluetooth listener

All peers have a Bluetooth Listener listening continuously for incoming Bluetooth communication requests. We implemented this Bluetooth Listener in the form of an Android service which runs in the background. The service runs until the application is running, and keeps listening for incoming communication requests.

Once a communication request is detected, it unbundles the information provided in the Bluetooth packet. This information contains the source address, the attributes extracted from the image, and timing and synchronization data. The knowledgebase is then checked to find a possible match. If such a match is found, it sends a message back to the sender containing information about the name of the matched object and its attributes. The sender then makes an entry into its own knowledgebase, and thus learns about the object from its peer.

2) Bluetooth query

When a device is unable to recognize an object, it initiates a Bluetooth connection with its peers, and sends a query with the attributes extracted from the current object image. Apart from the attribute information, the packet also contains the destination address and timing and synchronization information.

The contacted peer then tries to recognize the object based on the information present in its own knowledgebase. If it is unable to recognize the object, it sends back a negative signal back to the device who initiated the communication.

The source device attempts to query all its peers to find information about the current object. If any of the peer has information about the object, it makes the source device learn about the object i.e. it updates its knowledgebase.

Thus, if even one peer has information about an object which none of its peers know about, the entire system virtually has knowledge about that object, as any peer requiring knowledge about that object can learn from its peers transparently, whenever required.

V. Future Scope of the Project

AI, and the field of machine learning in particular, is a very vast field. It’s an exciting area with continuous research. Our project is just a modest entry-level effort in this direction. Hence, there are some things which we wish to take further. Some of the important ones are as follows:

- Using a cloud based system for the distributed AI will ensure that the system is not limited by any network range, and can connect with its peers from any location at any time.
- Implementing the system on on multiple platforms so that cross platform learning is also possible. This would enlarge the target range of the system so that it can be deployed on multiple platforms.
- The current shape detection algorithm uses a simple relation between the area of the object and the area bounded by the borders of the object. Though this works fairly accurately in ideal and close-to-ideal conditions, it behaves erratically in uneven lighting scenarios.
- Include more attributes, such as the absolute dimensions of the object, like length, breadth and radius. This would require multiple viewpoints to eliminate inaccuracies like parallax errors.

ACKNOWLEDGMENT

We are highly indebted to our project guide Prof. Avinash Shrivas from Vidyalankar Institute of Technology for his guidance, inspiration, constructive suggestion and encouragement that helped us throughout the evolution of this project. He has taken innumerable efforts to go through the project and make necessary corrections as and when needed.

REFERENCES

[1] H. Goyal and S. Yadav, "Multi-agent distributed artificial intelligence," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 1, June 2011.
[2] A. Correia, "Distributed artificial intelligence," Departmento de Engenharia Informatica, Instituto Superior de Engenharia do Porto.
[3] R.Duda and P. Hart, "Use of the hough transform to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11-15, January 1972.
[4] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape context," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509-522, April 2002.
[5] A. Diaz, P. M. L. Panizo, and A. M. Recio, "A survey on mobile peer-to-peer technology," in *Proc. XV Conference on Concurrency and Distributed Systems (JCSD'07)*, 2007, pp. 59-68.

Omkar Pimple completed his graduation in computer science and engineering from Vidyalankar Institute of Technology (affiliated to Mumbai University) in 2013.

He is currently working as a software development engineer at Mumbai Based Startup Gray Routes Innovative Distribution, which is engaged in the business of providing mobile-based solutions for the FMCG industry. His current interests include AI, machine learning and mobile development.

Umesh Saravane completed his graduation in computer science and engineering from Vidyalankar Institute of Technology (affiliated to Mumbai University) in 2013.

He is working as an Android development engineer at Gray Routes Innovative Distribution, Mumbai, which is engaged in the business of providing mobile-based solutions for the FMCG industry. His current interests include AI and algorithms.

Neha Gavankar completed her graduation in computer science and engineering from Vidyalankar Institute of Technology (affiliated to Mumbai University) in 2013. She is currently working at Network Intelligence (I) Pvt Ltd. Her current interests include information security and IT security protocols.