# Table of Contents

Abstract .................................................................................................................................................. 2

1. Need Identification .............................................................................................................................. 2

2. Objectives ........................................................................................................................................... 2

3. Requirements ..................................................................................................................................... 3

4. Research .......................................................................................................................................... 4

4.1. Overview of Object Recognition .................................................................................................. 4

4.2. Object Recognition and Robotics ................................................................................................. 6

4.3. Haar-like Features ......................................................................................................................... 6

4.4. Local Binary Patterns ................................................................................................................... 8

4.5. SURF ............................................................................................................................................. 9

4.6. Adaboost and Classifier Training ................................................................................................. 9

4.7. Positive Sample Generation Algorithm ....................................................................................... 10

4.8. BeagleBone Black ....................................................................................................................... 10

4.9. BeagleBone Black ....................................................................................................................... 11

5. Hardware ......................................................................................................................................... 12

6. Testing Environment .......................................................................................................................... 13

7. Conclusion ....................................................................................................................................... 16
ABSTRACT

The purpose of this research is to create a real-time object recognition system that works within the constraints of a robotic platform. This OpenCV-based Speaking Assistant Robot, called OSkAR for short, will accomplish two primary tasks. It will be able to create a database of object classifiers using feature-based methods, and then access that database for real-time recognition of objects in a real world environment. The methods used by the robot must be simple, quick, and accurate in order for it to accomplish its purpose, so this project focuses on improving existing object recognition methods to meet the needs of the system.

KEYWORDS: ROBOT, OPENCV, OBJECT RECOGNITION, HAAR-LIKE FEATURES, LOCAL BINARY PATTERNS

1. NEED IDENTIFICATION

One of the most vital and rapidly changing fields in computer science is the research and development of service robotics. With a population that grows larger and more diverse every single day, more and more opportunities arise where humans could surely use the assistance of robotic system. This is reflected in statistics gathered from the International Federation of Robotics. In 2011, 2.5 million domestic robots were sold, which is 15% more than in 2010. These types of robots vacuumed, mowed the lawn, or served some kind of entertainment purpose. Robots that assist the handicapped also suddenly became popular in 2011, jumping from 46 units sold in 2010 to 156. Although these are small numbers, this is predicted to be a major area in service robotics in the coming years, and research in the field is beginning to take off [IFR].

Within this field of service robotics, there is the problem of computer vision. In order to become human companions, these robots must be able to see the world as we do. They must have a system of vision that allows them to understand what they are viewing. There are a myriad of ways to go about doing this. This research, however, will focus on feature-based object recognition. The ability to tell one detailed object apart from another gives the robot a layer of intelligence that allows it to interact with the world in a human-like way. There are multiple ways to do feature-based object recognition, and many of them can be rather complex. The research presented in this paper will look at existing methods of object recognition, and then choose the most suitable method for integration into a mobile robotic system for the purpose of object recognition.

2. OBJECTIVES

This research project focuses on the development of an OpenCV-based Speaking Assistant Robot (OSkAR) that will incorporate an object recognition method into its programming. Therefore, the recognition process will purely cater to the needs of the robotic system. The project’s first objective is the creation of object classifiers, and the second objective is OSkAR’s use of these classifiers in a practical, assistive application. The robot must be mobile, and have the ability to navigate simple indoor environments. OSkAR must also be able to take pictures and video, and should have a way to notify its user when it views a recognized object. It should be noted that OSkAR will only be
able to recognize objects from one particular viewpoint, unless a view-invariant method is found in the course of this project. Most importantly, the robot’s classifier development and recognition must be both quick and accurate in order for it to be successful.

3. REQUIREMENTS

Most requirements for this project are related to the object recognition process. Feature-based classifier development begins from sample pictures. Both positive and negative samples are used to train classifiers. Positive samples contain the object of interest, while negative samples are just normal background images. Using an improved positive sample generation algorithm that will be explained later, the robot will be able to take pictures of an object, and then these pictures can be transferred to a more powerful computer to be processed into usable object classifiers. These classifiers are then cascaded in order of most important to least. Once this classifier cascade is created, the generated XML file can be uploaded to the robot. At this point, the robot can assist a human being by locating the object of interest. The primary focus of this project is the streamlining of the classifier creation process, and developing a method for running this process quickly and simply. Once again, feature-based recognition will be used in this project. Potential features types include Haar-like features, Local Binary Patterns (LBPs), and Speeded Up Robust Features (SURF). These types are used due to their support in OpenCV. OpenCV is a library of C++ programming functions for the use of computer vision. This project is also linked to personal previous research into positive sample generation, so classifiers from that research will be used in this project. The hardware for this project will include a microcontroller, a camera, motors, wheels, and any other materials needed to build the robot itself.

The requirements for this project are as follows:

- A mobile robotic platform equipped with a camera and voice system
- A group of 3 to 5 accurate object classifiers related to hallway recognition and navigation for use in a faster system of object recognition that also allows for multiple classifier use.
- A system capable of generating classifiers from large image datasets.

An objective tree was created using these requirements. The primary objectives are a mobile robot platform, a classifier development system, and a communication system between the robot and a controlling computer. This is shown in Model 1.
4. RESEARCH

4.1. Overview of Object Recognition

The first topic of this paper is object recognition in general. In order for a robot to interact with its environment, it must first be able to see its environment. However, it cannot optically view the world on the level of a human. The average adult human can look at a small assortment of simple household items, and quickly determine what each one is and how it is used. The most important tool in this scenario is the human’s brain. The brain is essentially a massive database of all the information the human has gathered in his or her lifetime. Additionally, we can see a broad range of items at one time, and all the 2D and 3D elements of those objects. Our ability to perceive the same object from multiple angles is also important [M. Bjorkman, 2006]. When approaching the topic of object recognition, one must consider both 2-dimensional and 3-dimensional objects. 2D recognition has already reached the stage where it can be used successfully in real-world applications. 3D recognition, however, is still being developed toward that goal. It has not been as heavily researched and it is filled with complex programming and hardware problems to solve. What clearly differentiates 3D from 2D is that 3D contains another plane. There is quite a bit more information to a description of a 3D object. 2D object recognition methods only have to solve the problems of translation and rotation. 3D objects, however, can be viewed from an infinite number of angles, and may look completely different turned one way than they do another way. Recognition methods must be devised that are capable of understanding and processing
such a large wealth of information [P. Viola, 2001].

The type of hardware used can also play a role in how an object is recognized. Sensors can be either contact or non-contact. Contact sensors physically touch and measure the object. Non-contact sensors, of course, do not touch the object. These include range sensors and camera setups. Non-contact sensing is preferred in industry since there is little to no chance of damaging an object, and it is less expensive and quicker than using moving parts with touch sensors [P. Viola, 2001].

There are also two types of vision systems: active and passive. In active vision, a light with a pattern is projected onto the object. Then, a camera gets an image of the object with the projected pattern. It can be quite useful with 3D objects, since the pattern will show contours, edges, and distance on the object. However, it is also a more expensive method. The passive vision method uses one or more cameras to get images of an object, and combines that with an algorithm for object recognition. This paper will focus almost entirely on non-contact, passive vision systems due to their lower expense and greater abundance of existing research [P. Viola, 2001].

When it comes to the actual identification methods of the objects, there are two approaches that can be taken. Some methods are generative, and others are discriminative. Generative models estimate the likelihood of an object existing, and then the object is assigned the most likely class. These models search to find a representation of the original data. Discriminative models discriminate in what is being looked at in order to find the object of interest. The purpose of these models is primarily for classification tasks. Using data from training, these methods try to reach optimal decision boundaries. When looking at an unknown sample, a decision boundary is estimated, and a corresponding label is assigned to the sample [P. Roth, 2008].

In object recognition, there are three main categories of features used for identification. First are appearance based features. These are usually taken from 2D images of the object of interest. Examples include color shade and intensity. It is easy to see where this method can only allow for very basic object recognition, like finding a green object among red objects. The second category is shape-based. These features represent the object by shape and contour. Third are model based features. Objects are approximately represented as basic 3D geometric shapes. These shapes include boxes, spheres, cones, cylinders, generalized cylinders, and surface of revolution [P. Roth, 2008].

Each method of recognition can be split into two main classes: local and global. A local feature is a particular detail located in a very small region, including even pixel-sized regions. It is a single piece of info distinctive to the object of interest. Local features include color, average gradient, and average gray values of pixels or small regions. When doing object recognition, local features should not be affected by illumination changes, noise, scale changes, or viewing direction. However, this is often not possible due to how simple these single features are. Therefore, several features of a single point are combined into a more complex description. This new image description is normally referred to as a descriptor [P. Roth, 2008].

Global features, on the other hand, look at the image as a whole. They try to consider all the pixels in the image. If necessary, the original image of the object can be reconstructed. This means, in some cases, object recognition can be very accurate. However when it comes to objects that may
only feature partially in an image, local features have the advantage. They can still recognize an object even if only a small part of it is actually visible [P. Roth, 2008].

4.2. Object Recognition and Robotics

Applying human visual skills to robotics is not a simple task. How does the robot differentiate one object from another, and subsequently define what it is? How does the robot recognize the same thing from multiple perspectives? The answer is all in the programming. A robot must be programmed to know what objects look like and what they do not look like. However, programming like this requires a bit of learning on the robot’s part. Feature types are used in this learning process. One of the more prominent and well-developed ways to create object recognition software is through use of OpenCV, or Open Source Computer Vision. OpenCV is a library of over 500 programming functions for the purpose of real-time computer vision. The library can be used with the programming languages C, C++, and Python. One of the major advantages of OpenCV recognition is its near-instantaneous speed of recognition, making it extremely useful in real-world applications [OpenCV Wiki]. In OpenCV, classifiers for recognizing different objects must be created by the user. An OpenCV installation already comes with a trained classifier for frontal face direction, but it is not limited only to that. Just about any object can be recognized if there is a classifier created for it. These classifiers depend on local features.

4.3. Haar-Like Features

Haar-like features are attributes found in images when looking at the intensities of pixel shades across areas. Several groups have used them for the purpose of object recognition in order to quickly find particular objects in an image [C. Papageorgiou, 1998][P. Roth, 2008]. Figure 1 shows the attributes and shapes of these features.

![Figure 1. Four different categories of Haar-like features, including 45° tilted versions added by Lienhart & Maydt[R. Lienhart].](image)

The features depicted in Figure 1 can be separated into different groups based on the number of rectangles involved. Two-rectangle features, seen in category 1 above, are the difference between the sum of the pixels in two horizontally or vertically adjacent rectangles placed on the image. These rectangles are the same size and shape. Category 3 also includes two-rectangle features, but they are the difference between the sum of pixels of one
larger rectangle and a smaller one centered inside it. Three-rectangle features, shown in category 2 above, compute the sum within two outside rectangles subtracted from the sum of a center rectangle. Then there is the four-rectangle feature, shown at the bottom of Figure 1. It computes the difference between diagonal pairs of rectangles [P. Viola, 2001]. Originally, the tilted haar-like features were not part of the basic set until they were tested and added by Lienhart & Maydt. The extended feature set complicated the learning process somewhat, but greatly improved recognition accuracy. The extended set produced a 10% lower false alarm rate when tested with face recognition. [R. Lienhart].

In a study done by Viola & Jones, a new, faster method of image representation is presented, called an integral image, and it is then used to rapidly find haar-like features within a given image. The study says that “an integral image can be computed from an image using a few operations per pixel. Once computed, any one of these haar-like features can be computed at any scale or location in constant time” [P. Viola, 2001]. The integral image at location x,y in the image contains the sum of the pixels above and to the left of x,y. It can be described mathematically in the following way:

\[ \text{ii}(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \]

In this summation, \( \text{ii}(x, y) \) is the integral image, while \( i(x, y) \) is the original image. Using the following reoccurring functions, the integral image can be computed in a single pass.

\[ s(x, y) = s(x, y - 1) + i(x, y) \]
\[ \text{ii}(x, y) = \text{ii}(x - 1, y) + s(x, y) \]

(\( s(x,y) \) is the cumulative row sum, \( s(x,-1) = 0, \text{ii}(-1,y) = 0 \))

Haar-like features are then generated through multiple array references. This can be explained using Figure 2.

**Figure 2.** Rectangular segments of an image.

The sum of any rectangle can be computed in just 4 array references. For example, in Figure 2, rectangle D can be computed in four array references.

\[
\begin{align*}
\text{ii}(1) &= A \\
\text{ii}(2) &= A+B \\
\text{ii}(3) &= A+C \\
\text{ii}(4) &= A+B+C+D
\end{align*}
\]

Therefore,

\[ D = \text{ii}(4) + \text{ii}(1) - (\text{ii}(2) + \text{ii}(3)) \]

The difference between two rectangular sums can be done with eight references, but adjacent rectangles only take six references. Three adjacent rectangles take eight, and four of them take nine references. [P. Viola, 2001].

As mentioned earlier, integral image representation is quick to compute. Real time calculation is important since many applications of object recognition have to be able to quickly and accurately recognize objects as soon as they see them. It is inconvenient to take pictures or video, and then scan the media piece by piece in order to find the object of interest. In our desired robotic object recognition system, the
looking and the recognizing should be occurring simultaneously.

4.4 Local Binary Patterns

OpenCV also supports Local Binary Patterns (LBP). Local Binary Pattern detection is an object recognition technique that has recently come into use, and it is becoming a more and more popular method. The method’s fast training time, decent accuracy, and speed of recognition is extremely convenient. The first step in recognition is the division of an image into small regions. The pixel-based LBP features are then generated from the regions, and placed into a feature histogram that represents their distribution. In this way, the texture of individual regions is saved, and these regional textures can then be used to recreate the general form of the image [H. Lian, 2007].

Essentially, the object images become compositions of micro-patterns. These patterns are invariant to changes in grayscale intensities. With all these local features, a global image can be created [H. Lian, 2007].

The simplest LBP is the 3x3 feature. A center pixel is selected, and then the neighboring pixels are converted to binary numbers. A pixel becomes a 1 if its value is greater than or equal to the center pixel. It becomes 0 if its value is less than the center. From this, a binary number string can be derived by moving clockwise from the top left pixel, and converted to decimal [H. Lian, 2007]. An example can be seen in Figure 3.

![Figure 3. An example of a 3x3 LBP represented by decimal number 40.](image)

There are a few commonly occurring feature types in images that can be seen in Figure 4.

![Figure 4. Types of LBPs [Bytefish.de]](image)

The algorithmic description of LBPs is as follows:

$$LBP(X_c, Y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

$(X_c, Y_c)$ is the central pixel with intensity $i_c$. $i_p$ is the intensity of the current neighbor pixel.
\[ s(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{else} \end{cases} \]

4.5 SURF

A third type of recognition is through the use of SURF, or Speeded Up Robust Features. Essentially, it is an improved version of SIFT (Scale Invariant Feature Transform). SURF recognition is both scale and rotation invariant, the latter giving it a major advantage over the other two feature types discussed in this paper. However, its recognition time is extremely slow. Real-time recognition is out of the question. It uses haar-like features, but rather than training classifiers with them, it calculates them on the fly using a single template image. This template is compared to either other pictures or videos, and connections are drawn between similar features. If there are enough parallels, then the object is considered detected. Additionally, because it is in-plane invariant, the object can be at any angle in the x-y plane [C. Evans, 2009].

![Figure 5. SURF recognizing a phone from a template using logos.](image)

SURF is particularly good at finding short words or logos in an image, as shown by Figure 5. However, since all of this calculation is happening before recognition can occur, SURF detection is sluggish and lags behind when examining videos. For real-time recognition, it is preferable to use a training process beforehand and pre-prepare the classifier.

4.6. Adaboost and Classifier Training

The problem with classifying objects using feature-based recognition is the sheer number of features that exist in an image. Therefore, an algorithm called AdaBoost (Adaptive Boosting) exists to train “weak” single-feature object recognition classifiers into stronger ones by using the most heavily reoccurring features in the set with the lowest chance of error.

- Given example images \( (x_1, y_1), \ldots, (x_n, y_n) \) where \( y_i = 0, 1 \) for negative and positive examples respectively.
- Initialize weights \( w_{1,i} = \frac{1}{m}, \frac{1}{2} \) for \( y_i = 0, 1 \) respectively, where \( m \) and \( t \) are the number of negatives and positives respectively.
- For \( t = 1, \ldots, T \):
  1. Normalize the weights,
     \[ w_{t+1,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{m} w_{t,j}} \]
     so that \( w_t \) is a probability distribution.
  2. For each feature, \( j \), train a classifier \( h_j \) which is restricted to using a single feature. The error is evaluated with respect to \( w_t \), \( e_j = \sum_i w_i | h_j(x_i) - y_i | \).
  3. Choose the classifier, \( h_t \), with the lowest error \( e_t \).
  4. Update the weights:
     \[ w_{t+1,i} = w_{t,i} \beta_t^{1-e_t} \]
     where \( e_t = 0 \) if example \( x_i \) is classified correctly, \( e_t = 1 \) otherwise, and \( \beta_t = \frac{1}{1-e_t} \).
- The final strong classifier is:
  \[ h(x) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, & \text{otherwise} \end{cases} \]
  where \( \alpha_t = \log \frac{1}{1-e_t} \).

The Adaboost algorithm for classifier boosting. [P. Viola, 2001]/[R. Lienhart].
For example, using haar-like features, every round of boosting selects the most-likely feature out of all the possible features in an image sub-window, and gives it more weight during a search. These features are essentially strung together from most common to least common, forming what is known as a classifier cascade [P. Viola, 2001]. In a classifier cascade, if the most common feature is found, then the program begins to search for the second most common feature. If that is found, it moves on to the third. This process repeats for a certain threshold level. At this point, the area containing these features is marked as an object-containing area.

Figure 6. Examples of common haar-like features found during facial recognition [P. Viola, 2001].

Figure 6 shows how faces can be recognized using just a few specific haar-like features. The two features shown above are the first and second most likely facial features picked by the Adaboost algorithm. The top row of Figure 6 shows only the features, and the second row shows the features overlaid in the corresponding areas on a face. As one may observe with most faces, the area across the eyes is much darker than the area across the upper cheeks. It can also be observed that the eye regions are darker than the bridge of the nose. These facial features are clearly demonstrated by the simplified rectangular features [P. Viola, 2001].

4.7. Positive Sample Generation Algorithm

In my own research prior to this paper, a particular method of positive sample generation was used. A method was developed that would increase recognition accuracy and simplify the positive sample collection process. Creating samples from one image meant there was a lack of variety, and gathering many positive images was a time-consuming task. Therefore, multiple unique vector files were generated for several different positive images, and then those files were merged together into one large vector file. The vector merging application used in this project was developed by Naotoshi Seo[13]. So, as an example, 10 pictures can be taken of an object. Those pictures are then edited so that they are completely transparent except for the object itself. Then, the first object image is placed onto random backgrounds multiple times and slightly distorted each time. This can be done 10-100 times for each single image, leading to 100-1000 positive samples.

4.8. BeagleBone Black

The method of object recognition is not the only important aspect of this project. OSkAR itself must have an on-board system that can run OpenCV with suitable performance. The BeagleBone Black was chosen as the center of this system. It is an inexpensive, credit card-sized Linux computer. It has an AM335x 1GHz ARM Cortex-A8 processor, 512 MB of DDR3 RAM, and can be programmed much like a microcontroller. It has 1 USB port, an Ethernet port, HDMI, an SD card slot, and 2x46 pin headers. By default it runs Ångström Linux, which will be sufficient for this project [Beaglebone.org]. Previous
work exists showing that OpenCV and the BeagleBone Black work well together, and that image processing can be done on the tiny system [D. Molloy]. This existing work also provides a starting point for what must be done in this project.

Figure 7. Close-up of the BeagleBone Black (BBB)

One issue that may need to be solved is optimizing the BBB for real-time video streaming. Luckily, documentation exists online explaining how to achieve 30fps with the board, OpenCV, and a Logitech C-series webcam. NEON hardware acceleration is available for use on the BBB, and OpenCV must be rebuilt with NEON enabled. Cross-compilation is used to complete this process. A more optimized JPEG codec should also be installed on the BeagleBone [M. Darling].

The BBB will need to be powered from a battery source in the final project, so amperage and voltage requirements must be determined for all the devices connected to it. A battery can then be selected that will be able to power all the connected items. Power must be routed primarily to the camera, the USB Hub, and the BeagleBone Black.

4.9 Mobility and Navigation

The mobile platform of the robot will be a secondary task in this project. Once the camera and BeagleBone Black board function as a recognition system, they can be placed on any mobile platform for use. The mobile platform we plan to use in this project will consist of a plastic base that drives on motor-driven wheels. A possible collision detection system may be implemented using distance sensors. All navigation will be handled by an Arbotix microcontroller.

Figure 8. Arbotix Robocontroller with Xbee & Commander
This board features:
- 16MHz AVR microcontroller (ATMEGA644p).
- 2 serial ports, 1 dedicated to Bioloid servo controller, the other to the XBEE radio
- 32 I/O, 8 of which can function as analog inputs
- Servo style 3-pin headers (gnd, vcc, signal) on all 8 analog inputs, and 8 of the digital IO
- Dual 1A motor drivers, with combined motor/encoder header.

(http://www.vanadiumlabs.com/arbotix.html)

5. HARDWARE

The following items will be used to construct the initial prototype of OSkAR.
- BeagleBone Black ($55)
- Power Supply AC Adapter ($7.51)
- USB 2.0 4-Port Hub ($14.00)
- BeagleBone Black Case ($17.95)
- External Sound Box ($9.15)
- Logitech HD Pro Webcam C910 ($145.00)

The following will be used to create and control the mobile platform.

- Arbotix Robocontroller ($39.95)
- Arbotix Commander ($59.95)
- Robot DC Gearhead Motor & Wheels Kit ($89.95)
- Xbee Modules ($19.00)
- IR Distance Sensors, 20cm-150cm ($15.95 each)

OSkAR’s object recognizing system will follow these steps:
1. Boot-Up via Switch
2. Begin Object Recognition Program
3. Begin taking camera input
4. Analyze each received frame for objects of interest
5. Report name and location of objects seen
6. Continue Operation until User Shutdown via Switch

The mobile platform for OSkAR will consist of an Arbotix Robocontroller ($45), Motor-driven wheels, and a controller that communicates with the Arbotix board via Xbee Serial communication. The user will drive the robot via the controller. It is possible that IR sensors will be used for override autonomous navigation.

Figure 9. Preliminary sketch of OSkAR
All code for the BBB will be cross-compiled from a Linux netbook, then uploaded to the board itself to run. Cross-compilation is necessary since the board uses ARM architecture, but the Linux netbook uses x86. The purpose of this is to compile code faster and with more efficiency. Classifier Training will take place on a separate Windows 7 machine.

6. TESTING ENVIRONMENT

The testing environment for this robotic system will be extremely controlled. A university hallway will be used as the location. Various objects and structures within the hallway will be recognized by the system to assist the user of the robot. These objects may include, but are not limited to:

- Door Frames
- Stairway

Figure 10. Primary parts of OSkAR’s system, and their connections with the BeagleBone Black in the center.
Windows
People (Upper Body Detection)

The robot will be driven by the user down this hallway. As the robot drives, it will report what it sees out loud to the user. So upon seeing a nearby object, it will state the name of the object.

The robot must have an operation error rate sufficiently low enough to allow for proper operation. Therefore, the number of false positives, missed recognitions, and accurate recognitions during its journey down the hallway will be tallied up and analyzed. In our prototype, we will aim for a success rate greater than 75%, which will be calculated using the number of correct recognitions divided by the expected number of recognitions. Basically, if there are 4 doors in a hallway, and the robot only correctly identifies 3, that is a 75% success rate. However, False Positives must also be recorded to determine usefulness of the classifier. False positives are when the program identifies an object that is not actually there.

To test the testing scenario, a preliminary classifier was created using a small dataset of 100 positive samples and 1000 negative samples. 10 stages of training were allowed. This was tested using a simple laptop webcam.

A second classifier was created using an even smaller dataset. This had 100 positive samples again, and 500 negative samples. 15 stages of training were allowed.

The door corner feature already looks like a promising way to detect the locations of doors down a hallway. Figure 13 and Figure 14 show the positive samples and their randomization.
Figure 13. Examples of positive input samples with black backgrounds

Figure 14. Generated Positive Sample Examples.

Table 1. Timeline of Project

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify project requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify goals &amp; steps of project</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify Deliverables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do survey of literature and research</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write Paper</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prepare final presentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Present Project Plan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install OpenCV on BagelBone</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop Classifiers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implement recognition program onbag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add classifiers to BDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Distance Output System</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Inferred Platform</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test And Improve</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write Final Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Present Final Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### First Parts List

<table>
<thead>
<tr>
<th>Catalog number</th>
<th>Quantity</th>
<th>Item Description</th>
<th>Unit Price</th>
<th>Total Price</th>
<th>Date Received</th>
<th>B/O to date</th>
</tr>
</thead>
<tbody>
<tr>
<td>B00CHYOLKH</td>
<td>1</td>
<td>BeagleBone Black DevKit</td>
<td>$55.00</td>
<td>$55.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B002FA5WE2</td>
<td>1</td>
<td>Power Supply AC Adapter 5V 2.5A for D-Link</td>
<td>$7.51</td>
<td>$7.51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B003MTTJ0Y</td>
<td>1</td>
<td>Edimax EW-7811Un 150 Mbps Wireless 11n Nano USB Adapter</td>
<td>$8.99</td>
<td>$8.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B004OBZ088</td>
<td>1</td>
<td>Gear Head USB 2.0 4-Port Hub with Energy Saving Switch</td>
<td>$14.00</td>
<td>$14.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B00EO7JYTS</td>
<td>1</td>
<td>Expandable BeagleBone Black Case</td>
<td>$17.95</td>
<td>$17.95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B002R33VW</td>
<td>1</td>
<td>Sabrent External Sound Box USB-SBCV</td>
<td>$9.15</td>
<td>$9.15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SUBTOTAL:** $112.60

**SHIPPING & HANDLING (approximately 10%):** $11.26

**TOTAL:** $123.86

### Table 2. Order Form of Parts

7. **CONCLUSION**

This project detailed in this proposal will be created during the Spring 2014 semester. The work will be spread out over the course of the semester. Programming will be the first step, and hardware will be implemented second. The end result of the project should be a robotic assistant that can recognize objects in a school corridor and speak their names out loud.
REFERENCES