The Duckling Problem and Foreground Detection and Classification of Objects

CS88/188

Geethmala Sridaran
Nimit S. Dhulekar

TABLE OF CONTENTS

1. ABSTRACT:

“We go where our vision is” – Joseph Murphy, *prosperity writer extraordinaire*

We present a supervised technique for solving two computer vision problems; namely the Duckling Problem and detection and identification of objects. The motivation for this project was the requirement of implementation of a process to collect training data to measure the efficacy of the brain-derived vision algorithm being developed in the Neukom Institute. Identifying and learning the movement of humans from sensor data opens up a wide range of applications. It explores the idea of a robot being able to identify a particular person and also follow him or her around. This part of the project is referred to as the “Duckling Problem” because it creates a scenario where the robot follows a human which can be compared to a duckling following its mother. The two tasks might seem intrinsically different but for the purpose of the main goal, they can be combined into one integrated process.

2. Introduction:

The Duckling problem is designed to help a person control the robot without using remote controls. In order to facilitate this, the robot should be able to identify a person whom it should follow and then follow him/her around. The challenges one could see immediately here is the recognition of the person and to track the right person. In order to facilitate recognition, SIFT (Scalar-Invariant Feature Transform) algorithm is used in this project and many different tracking strategies were implemented which is explained in the following sections.

The object identification task is one of the most crucial problems in computer vision today. Primary for detecting an object is to be able to detect its presence in a clutter. Imagine having to identify a toy in a child’s bedroom which is full of toys. This problem is not very difficult for humans but poses a considerable challenge for present low-processing power robots. We have tried to implement a solution by which we can make it slightly easier to identify objects even when placed in a clutter. The freedom we allow ourselves is using a laser to actually point out an object. In case the object exists in BrainBot’s database, it can identify the object. If the object is something new, BrainBot will go around the object taking different perspective views of it and storing these views in its database. The purpose of taking different perspective views is to allow BrainBot to identify the object at a later time, even if approached from a different angle.

3. Experimental setup:

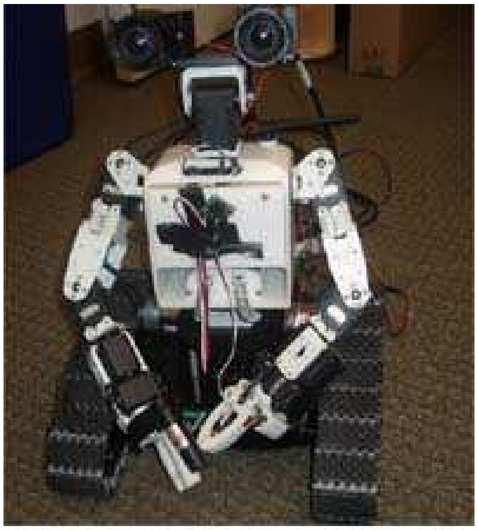


Figure 1

BrainBot [1] is a project directed and funded by the Brain Engineering Laboratory and Neukom Institute at Dartmouth College, whose purpose is to enable the study and practice of brain engineering.

BrainBot has multiple sensors attached to it. The primary sensor is the cameras. We are using the Prosilica GC650 cameras. The camera is well-suited for applications requiring speed and excellent image quality. Also the Prosilica cameras come with a software viewer called GigEViewer which allows us to see what the cameras are seeing.

The output from the cameras is in the widely adopted “Bayer” pattern which is a repeating 2x2 arrangement like GRGR… then BGBG... This output is called sequential RGB and allows for better compression of images. The difficulty with the “Bayer” pattern is that it only has one filter per pixel thus it only outputs one colour per pixel. The other 2 colours in the particular pixel need to be interpolated. We are using a simple probabilistic k-nearest neighbour algorithm for this purpose.

BrainBot also has a speech engine with a TechLink Speaker. The speech system allows 2-way communication, thus we can talk to him and he can answer back. We are using the Dragon Naturally Speaking software for communication with him. We also installed the MaxBotix MaxSonar-EZ4 High Performance SONAR Module for depth perception. We realized very late that this particular SONAR module can't be used for depth perception with soft targets like humans. We are moving to the EZ0 and EZ1 which is supposed to work better with humans.

BrainBot is a tethered robot and is constrained by its tether to the server laptop which is used to communicate with it. The Prosilica cameras are also connected to the laptop running the code via Ethernet cables. As part of future work, we are going to put the code on an x-86 processor which will be on BrainBot itself and have wireless connection for the Prosilica cameras.

4. Background:

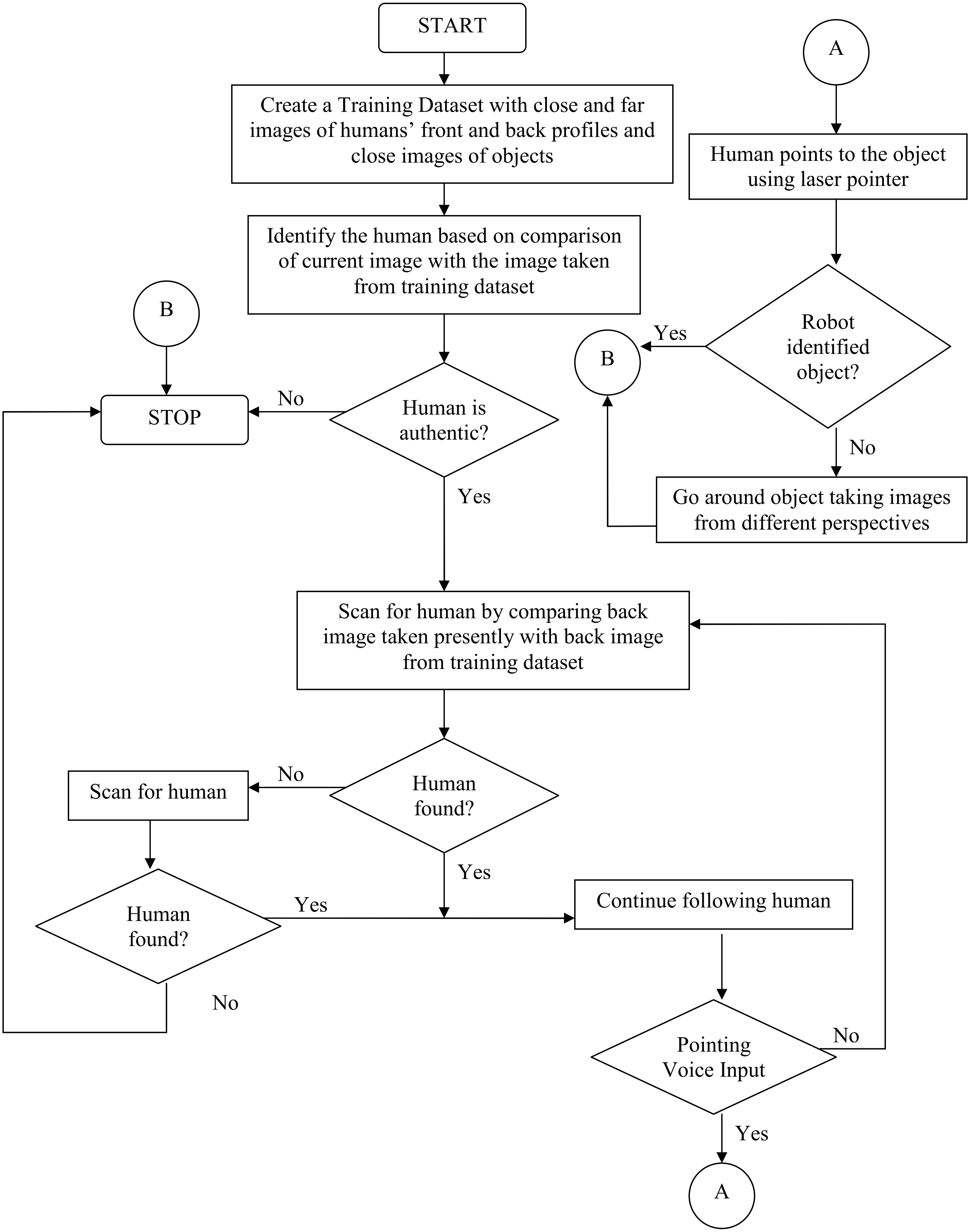
There are various face recognition algorithms [3] [4] and more in the field of computer vision. The reason for using SIFT here is to maintain simplicity and yet achieve promising results. To the best of our knowledge, this is the first time SIFT is being used to identify people so extensively. SIFT has the advantage of being invariant to illumination and orientation. Let us briefly go through the SIFT algorithm and its implementation [2].

SIFT (Scalar-Invariant Feature Transform)
SIFT helps in identifying local features in an image. The algorithm was published by David G. Lowe. The algorithm consists of the following steps.

1. **Scale-space extrema detection:**
   In this the interest points called the keypoints are identified. For this image is convolved with Gaussian filters and the difference of successive Gaussian-blurred images are taken. Keypoints are then taken as maxima or minima of the difference of Gaussians.
2. **Keypoint localization:**
   There are many keypoints obtained from step 1 of which some are unstable. These can be eliminated by performing a detailed fit to the nearby data. This is achieved using interpolation of nearby data for accurate position. Then discarding low contrast key-points and eliminating edge responses.
3. **Orientation assignment:**
   Each keypoint is assigned one or more orientations based on image gradient directions.
4. **Finding the keypoint descriptor:**
   The feature descriptor is computed as a set of orientation histograms on (4 x 4) pixel neighbourhoods. The orientation histograms are relative to the keypoint orientation and the orientation data comes from the Gaussian image closest in scale to the keypoint's scale. Histograms contain 8 bins each, and each descriptor contains a 4x4 array of 16 histograms around the keypoint. This leads to a SIFT feature vector with (4 x 4 x 8 = 128 elements). This vector is normalized to enhance invariance to changes in illumination.

5. Algorithm and Implementation:

The main algorithm that we are following is illustrated in the flowchart given below:


START
Create a Training Dataset with close and far images of humans' front and back profiles and close images of objects
Identify the human based on comparison of current image with the image taken from training dataset
B
STOP
No
Human is authentic?
Yes
Scan for human by comparing back image taken presently with back image from training dataset
Human found?
No
Scan for human
Human found?
Yes
Yes
No
Continue following human
Pointing Voice Input
No
Yes
A
A
Human points to the object using laser pointer
Robot identified object?
Yes
B
No
Go around object taking images from different perspectives

The following block diagrams give an extensive overview of the Duckling/Object Identification implementation:

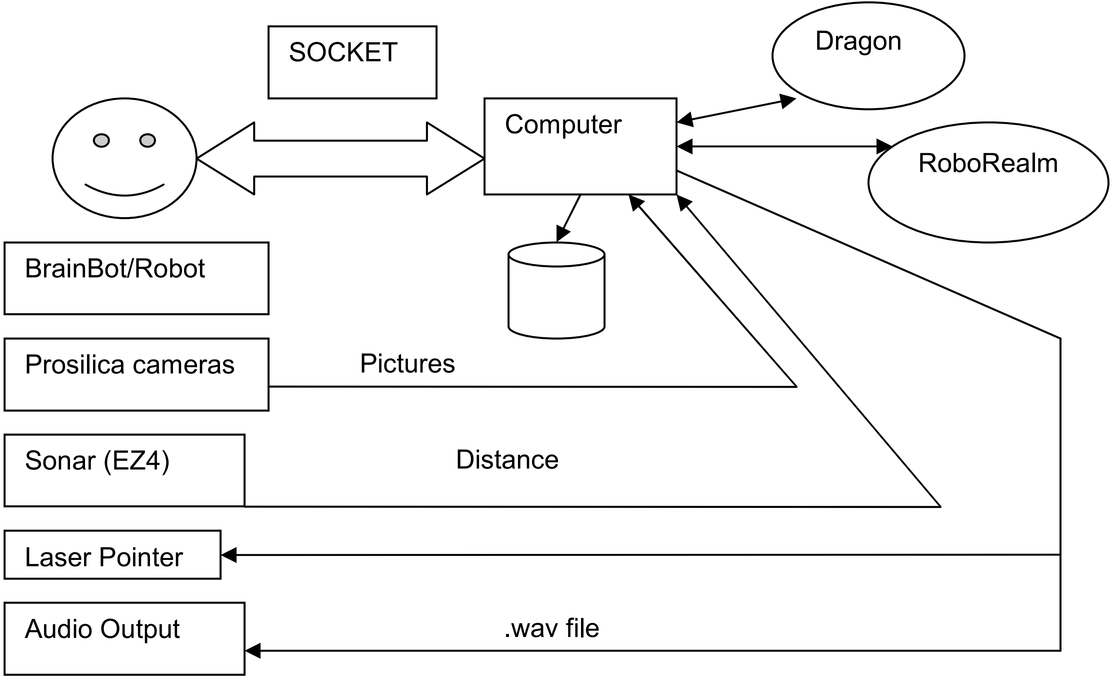


Figure 2: Main components used in the experiment

BrainBot communicates with the computer using socket (refer to sFigure 2). The Prosilica cameras take the images and store them in a hard disk connected to the laptop. Our earlier implementation used CAMERA 2.4Hz COLOR CCD Camera. But those cameras had a lot of static noise which was overcome by the Prosilica cameras. Before giving to SIFT, the image was converted to grayscale. This compensated for the fact that SIFT has a problem matching clothing. Also it was assumed that the person is in the centre of the frame so that the rest of the image can be blacked out. The Duckling program uses SIFT on this image and does a match to the image in the training data set based on k-nearest neighbour (knn) algorithm and Best-Bin-First (BBF) approximation algorithm to increase the computation speed. The implementation provided by Robert Hess [5] has been modified to suit the purpose of the Duckling problem. If a match (number of matches greater than or equal to 14) is found, the robot asks the person to turn around so that the person’s back can be captured and is stored in the hard disk for future matching. The robot speaks using a .wav file sent over a socket from the computer. If a match is not found the robot starts scanning to search for the person.

When a successful match is found, BrainBot begins to move. A feature was also added where BrainBot voiced the name of the person. Every three seconds it takes an image of the person to check if the person is still in the view or it has lost him/her. If the SIFT match fails, BrainBot stops and starts scanning for the person. For the back, images were not converted to grayscale as it was assumed that the person will wear the same clothes while being followed. All the commands to the robot are sent in ASCII format over the TCP socket.

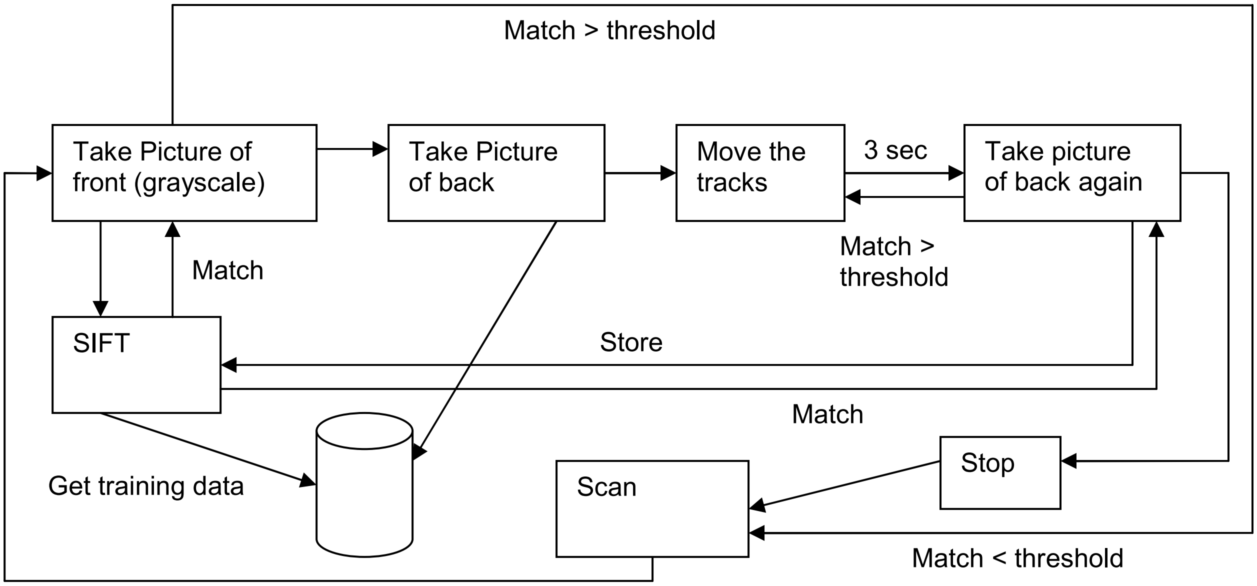


Figure 3: The Duckling Problem implemented with camera input

In order to get better results about the depth perception, sonar was mounted on BrainBot. Every three seconds the sonar would fire and get the distance of the person from BrainBot. If this distance is too close or too far a message would be sent out alerting the person to move far or to wait up respectively. If the distance was greater than a lower threshold (T1 = 50) and lesser than an upper threshold (T2 = 200), the robot will move. Adding a sonar module in Figure 3 will result in Figure 4 as shown below.

In addition, voice input was also given using the Dragon Speech to text converter. For this a script implemented by Karn Seth, in AutoIt, is used. Using this, BrainBot asks if it is moving in the right direction every three seconds. Based on the user's reply BrainBot decides whether to follow or to stop. There was also a text based implementation for the same. The implementation order consisted of checking the sonar module first, and then the voice/text input and lastly the picture matching. This was done since the former two were computationally much faster as compared to the latter. Also, voice input and text input can be given to switch from the Duckling problem to the object identification problem.

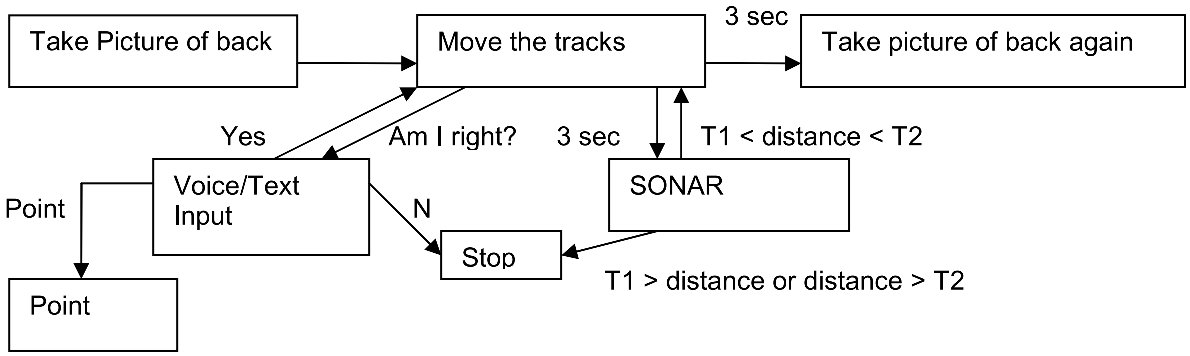
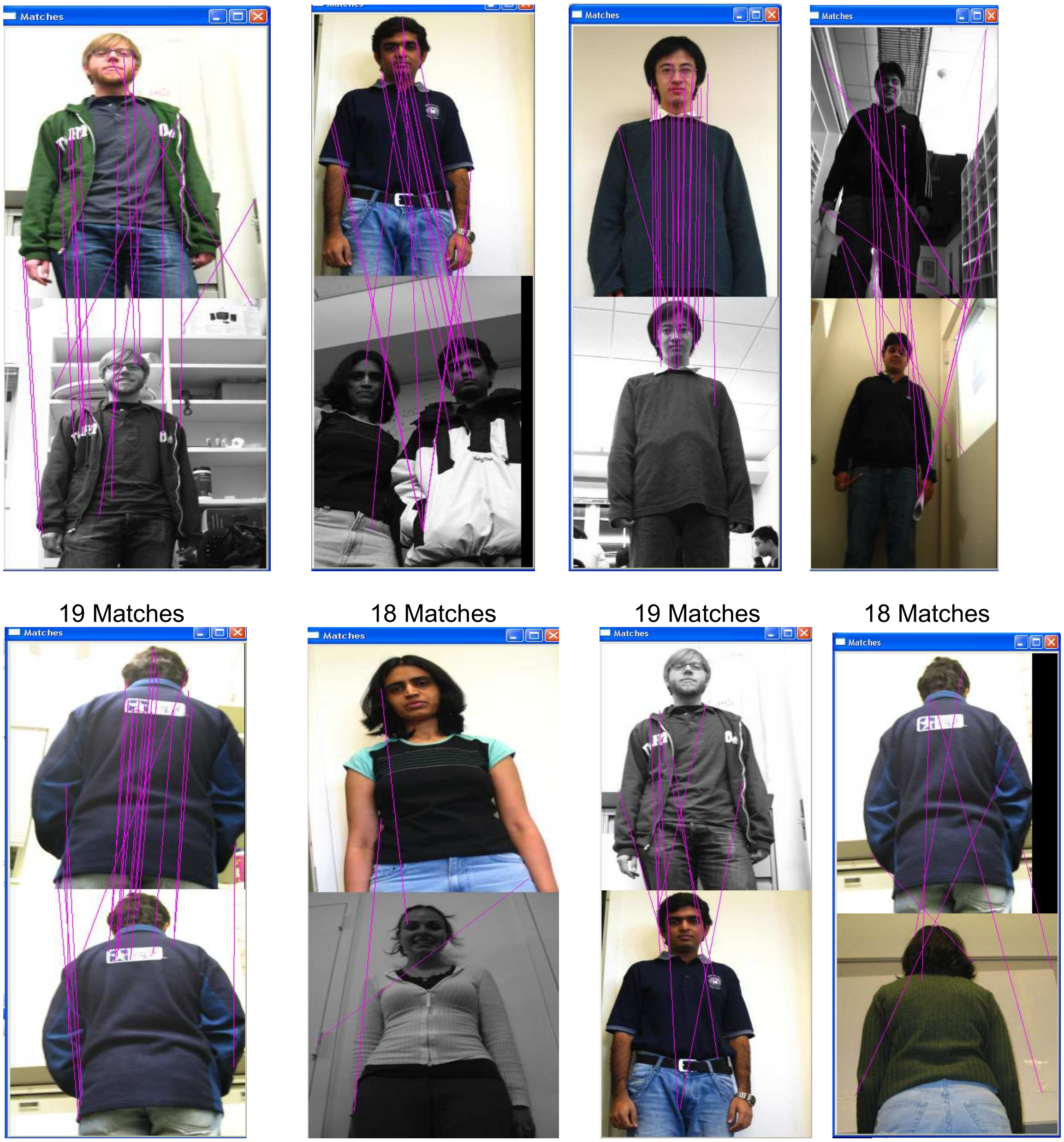


Figure 4: The Duckling Problem implemented with SONAR and Voice input

6. Results:

6.1 Duckling Problem:

The SIFT algorithm proved to be really successful in finding the same person in a cluttered environment and in distinguishing between people. Figure 5 helps in visualizing this fact. The threshold value is 14 and any greater value would be considered a success.



19 Matches 18 Matches 19 Matches 18 Matches

20 Matches 4 Matches 11 Matches 7 Matches

Figure 5: SIFT algorithm on different people

The algorithm got about 19 true positives and 3 false positives and 1 false negative for an overall of 23 test runs. It usually failed to recognize separate persons from the images of their backs in case they are wearing similar clothing. But in general it proved to be robust.

The SONAR module did not perform well with the test setting because the sensor used was EZ4 and this failed to pick up soft targets like humans after a distance of about 18 inches although, it picked up flat objects like wooden boards at a distance of about 72 inches (refer Table 1).

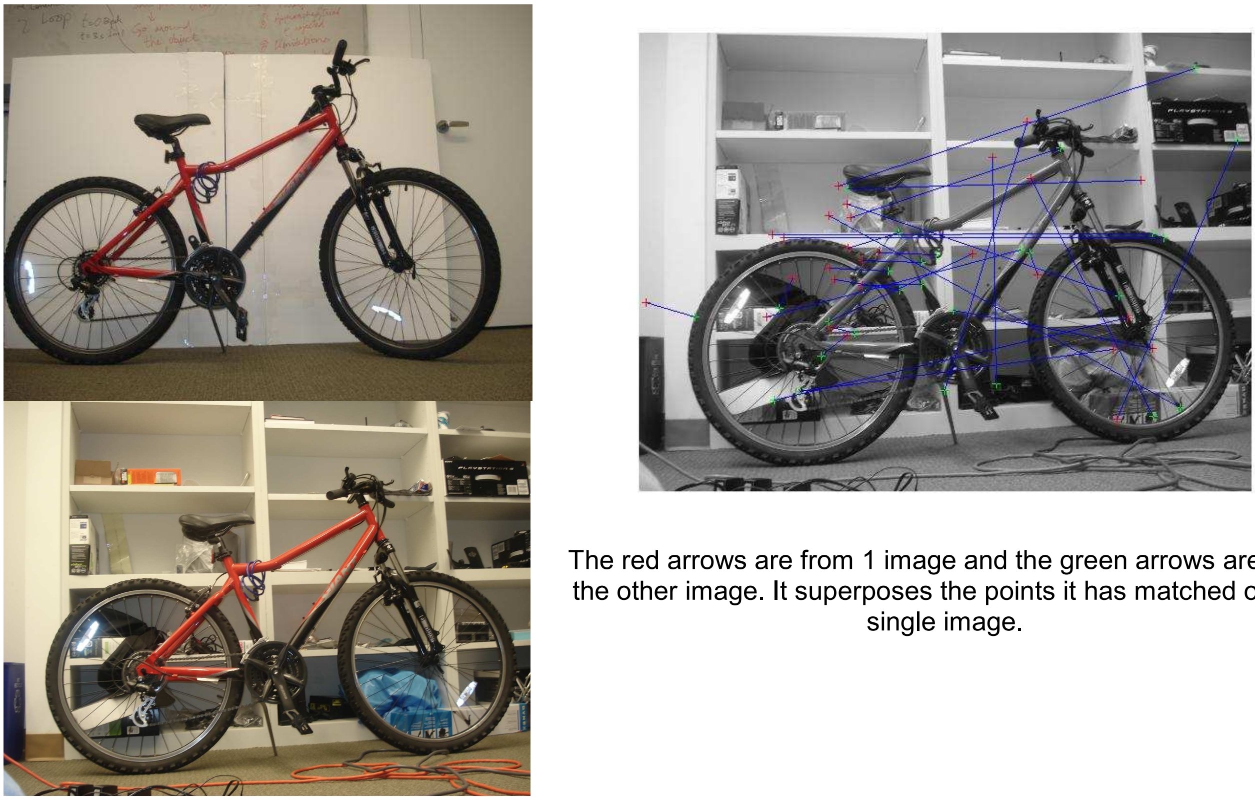
SONAR Values for a person	SONAR Values for an object	Distance (Inches) For a Person
11	11	6
30	29	12
53	45	18
266	60	24
390	95	30

Table 1: Sonar values for a person and object

Also, the voice input from Dragon software is not always reliable because of the background noise. To allow BrainBot to not be too dependent on it, the code has been given macro protection turning on the voice input only in case of emergencies like other failure of other sensors.

6.2 Object Identification and Classification Problem:

The object identification problem has been tackled with many different algorithms. One of the first algorithms we tried was Random Sample Consensus (RANSAC) [6]. The RANSAC algorithm estimates parameters for a mathematical model taking into account outliers present in the observed data. The performance of the algorithm improves as the number of iterations increases. We used the Harris corner detector algorithm [7] to generate corners for the various objects present in the image. We used MATLAB code from [8] as a preliminary test to verify whether the RANSAC algorithm would work for us. The results as can be seen from figure 5 were not very encouraging and it matched the particular object to random objects in the background. Thus we rejected this approach.



The red arrows are from 1 image and the green arrows are from the other image. It superposes the points it has matched onto a single image.

Figure 6: The training image, the cluttered image and the result after running RANSAC algorithm.

The second approach we tried was rather unique. It was based on the intuition that an object in the foreground would stand out from an object in the background. We found code from [9] which generated disparity values and confidence values for each pixel from 2 stereo vision images (refer figure 7). Since we are using 2 cameras, we could provide the software with the requisite images. We considered threshold values X and Y such that for each pixel

```
If (Confidence value > X and Disparity value < Y)
        Keep the original image pixel
Else
        Blacken the pixel
```

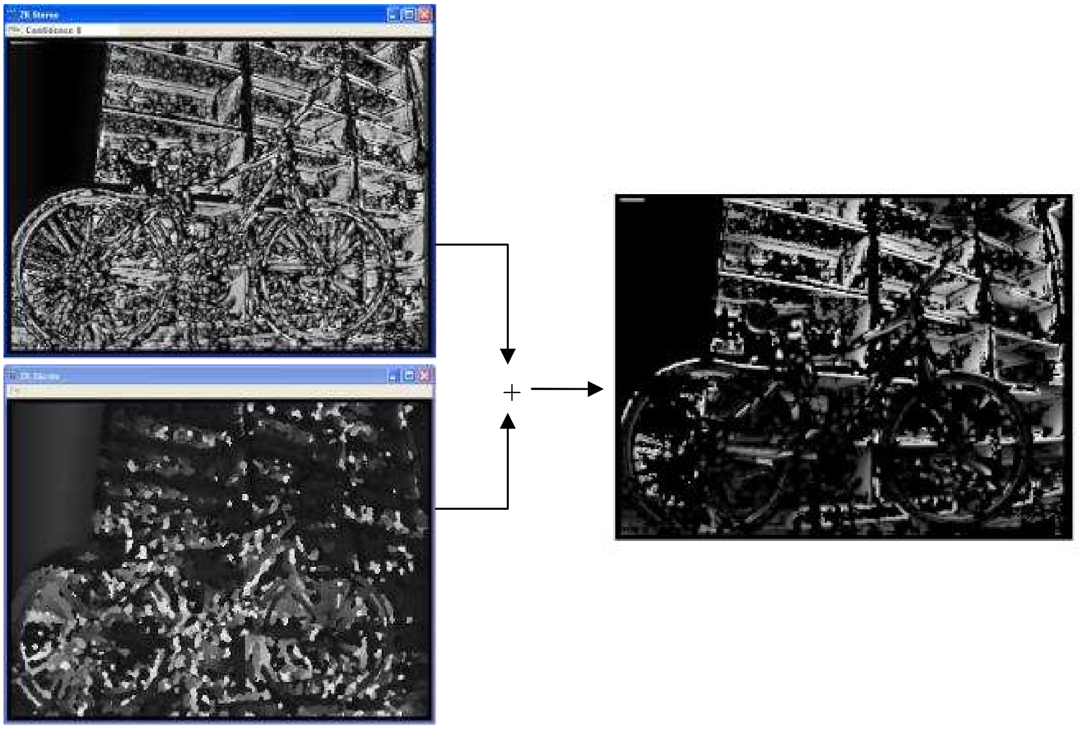


Figure 7: Implementation of Zitnick Kanade Algorithm

We tried various threshold values to see which returns the best results. The results were again not very acceptable. There were 2 reasons that this approach failed to produce good results. The algorithm required the foreground objects to be quite distinct from the background and we didn't want to constrain ourselves with the positioning of the objects. Also the algorithm required a long time to run, on the order of 30sec for the default setting of 8 iterations of refinement of the disparity map. Increasing the number of iterations to further refine the disparity map increased the processing time by about 5-10 sec per iteration thereby making it totally inadequate for real-time processing (refer to Table 2).

Number of iterations	**Time taken (in sec)**
5	21
6	23
7	26
8	30
11	44
12	50
14	60
15	66

Table 2: Performance measure for Zitnick and Kanade Cooperative Stereo Vision Algorithm

The third approach we tried was based on the paper [10]. The paper proved that the SIFT algorithm [2] combined with Hessian-affine feature finder was robust to viewpoint change. Although the experimental setup defined in the paper was different from ours, the main idea remains the same. They have used objects placed on a rotating tray and capture images from cameras placed at different heights and we are using stereo vision to look at stationary objects. The results were highly encouraging and are presented in Table 3.

Object type	Correct matches	Incorrect matches
Metallic Box	5	1
Bunny	0	1
Cardboard Box	6	2
Telephone directory	25	1
Graphics Card Box	20	1
Playstation 3	5	1
Robot	8	0
Seal	6	2
Stool	2	7
Toolbox	0	1
Tripod	20	1

Table 3: Performance measure for SIFT

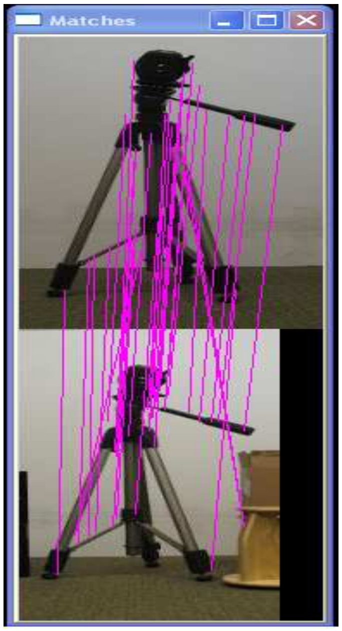


Figure 8: Matching the training image of a tripod stand with the tripod stand in a clutter.

To improve the performance of the SIFT algorithm, we wanted to make it slightly easier to detect the particular object we want to identify. Thus we included pointing out the object's top left and bottom right corners using a laser pointer. This allows us to create a minimum bounding box around the object in question. The challenge was to be able to identify the laser pointer. To do this we took 30 images per second and captured images for about 5 seconds. For the first second, we didn't shine the laser and then we did. Thus there was an obvious difference in the intensity values for the pixel where the laser shone. We could then average out the pixel value for each pixel in the image and then take the standard deviation of the average from each pixel. The pixel which had the maximum deviation was selected and its coordinates were padded with 15 pixels on each side to allow for human error in pointing out the exact location of the corners. If the SIFT algorithm runs successfully, BrainBot will be able to identify the object. If it is unable to identify the object, it goes around the object taking different perspective views of it.

7. Applications and Future Work:

Once the tracking and recognition is achieved, one can use this for applications e.g. follow a child and alert the parent when a child falls down or crosses a certain boundary. This idea can also be extended to patient care in hospital and probably a seeing-eye dog implementation. The object identification problem will build the training dataset for the under-development vision algorithm. Also future applications could range from the robot opening a door to crossing the road itself without any human intervention.

8. Conclusion:

In conclusion, our project has achieved identifying and tracking of humans and objects. This opens the door to plenty of applications in real world scenario. We would not be surprised if the robots are able to help out people in carrying out their daily chores and other activities.

(I) Acknowledgements:

We would like to thank the following people for their support and valuable suggestions to this project

1) Professor Tanzeem Choudhury
2) Professor Richard Granger
3) Professor Andrew Felch
4) Ashok Chandrashekar Holla
5) Karn Seth
6) Matthew P. Elkherj

We would like to extend special thanks Jon Hylands for his support in debugging the BrainBot.

We would like to also thank the volunteers for the Duckling problem
1) Anuj Bhatt
2) Christine Claudino
3) James M. Hughes
4) Mu Lin
5) Nancy Wendlandt

(II) References:

1. BrainBot URL http://www.bioloid.info/tiki/tiki-index.php?page=BrainEngineering+BrainBot
2. David G. Lowe, "Object recognition from local scale-invariant features". *Proceedings of the International Conference on Computer Vision 2*: 1150–1157
3. H. Moon, P.J. Phillips, "Computational and Performance aspects of PCA-based Face Recognition Algorithms". *Perception*, Vol. 30, 2001, pp. 303-321
4. A. Bronstein, M. Bronstein, and R. Kimmel, "Expression-invariant 3D face recognition, Proc. Audio & Video-based Biometric Person Authentication (AVBPA)". *Lecture Notes in Comp. Science 2688*, Springer, 2003, pp. 62-69
5. Rob Hess, School of EECS @ Oregon State University "SIFT feature detector" http://web.engr.oregonstate.edu/~hess/
6. M. A. Fischler and R. C. Bolles (1981), "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". *Communications of the ACM*
7. C. Harris and M. Stephens (1988), "A combined corner and edge detector". *Proceedings of the 4th Alvey Vision Conference*: pp 147--151.
8. Peter Kovesi "MATLAB and Octave Functions for Computer Vision and Image Processing". http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/
9. C. L. Zitnick and T. Kanade, "A Cooperative Algorithm for Stereo Matching and Occlusion Detection". *Proc. IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 7, July 2000
10. Pierre Moreels and Pietro Perona, "Evaluation of Features Detectors and Descriptors based on 3D objects". *International Journal of Computer Vision*, 2007
11. David G. Lowe's implementation of SIFT "Demo Software: SIFT Keypoint Detector" http://www.cs.ubc.ca/~lowe/keypoints/