International Journal of Advances in Computer Science & Its Applications– IJCSIA
Volume 5 : Issue 2 [ISSN : 2250-3765]
Publication Date: 30 October, 2015

Android Based Autonomous Arduino Bot

Aninda Saha, Shuvo Kumar Paul, Mahady Hasan and Md. Ashraful Amin

***Abstract*— This work reports the design, construction and control of a two-wheel object tracking robot. The Android controlled Arduino bot is a robot armed with a brain to recognize and track objects persuasively. Object recognition and tracking were achieved through extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object. However, we have also demonstrated tracking through offline data because of the limitations of the processing power of the cellular devices.**

***Keywords*—Detection, Tracking, Arduino bot**

I. Introduction

In the past decade, mobile robots have contributed its splendour in the military and industrial settings, as well as stepped into civilian and personal spaces such as hospitals, schools and homes. Having variety of purposes, robots are built to chip-in almost all spaces. In this paper a robot model on wheeled platform is designed which is capable of object recognition and tracking. The challenges in tracking evolves when the target objects change their appearance and shading conditions. The tracking methods used here are intended to overcome these main challenges.

To manage the robot system the following equipment was selected: microcontroller Arduino, to control the actuators of robot having acted like the brain of the system; motor shield driver, to control the DC motors; power module which reads the control signals from the microcontroller and turns on the appropriate motors power; wireless module, which is used for communication with the mobile terminal and an Android device to process frames for recognition and tracking.

In moving Target Classification and tracking from Real-Time Video [1], object-tracking algorithm combining the temporal differencing and template matching were used whose main drawback was misclassification can occur if multiple targets are close together and that small human targets are often not recognized as temporally stable objects.

The W4 algorithm [2] can analyse motion and track through occlusions, however the algorithm fails under drastic scene illuminations since its algorithm in separating foreground from the background is an intensity-based method.

Besides, Real-Time Human Motion Analysis by Image Skeletonization [3] presents a way to classify the motion of human target in a video sequence. Moving targets are detected and their boundaries extracted. But the weakness lies that the object of interest must move in a straight line and in a consistent direction, otherwise it'll be rejected as background.

A different number of tracking algorithms were reviewed in the literatures. The trackers demonstrated in this work are kernel-based that have been quite illustrious due to its simplicity and robustness to track variety of objects.

A. Saha, S. K. Paul, M. Hasan, and M. A. Amin
Computer Vision and Cybernetics Group
Department of Computer Science and Engineering
Independent University, Bangladesh (IUB)
Dhaka-1229, Bangladesh
www.cvcrbd.org

II. Implementation

The main aim of our work is to build a Arduino-based bot integrating with Android device that would be responsible for detection and recognition of an object that is being learned. The Android would pass processed command to the Arduino to make the bot move and track in accordance with the position of the object.

Constructing the bot is what comes first in the pipeline of our project which would eventually be tracking objects. For our work, we needed to control two motors, the distance sensor and react to remote commands coming over Bluetooth all at the same time. In future this could also be extended to add more functionality since the possibilities are endless. The initial workflow is as follows:

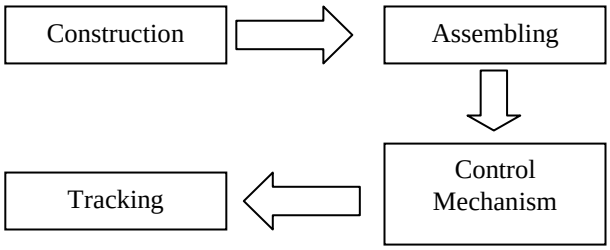


Figure 1. Project workflow

A. *Construction*

We have used ATmega328 based single-board microcontroller Arduino Uno R3 as the brain of our bot which would be making decisions in controlling the wheels for the purpose of tracking/following. The board is power sourced with a 9V battery and we have made use of its digital I/O pins for integrating the remaining components.

The Arduino board cannot directly control the motors of the bot, as it involves selectively running forward or backward that requires swapping power and ground inputs. A specialized circuit called an H-Bridge is required to make this happen and we have assigned this task to a motor driver that is basically a current amplifier which takes a low-current signal from the microcontroller and gives out a proportionally higher current signal which can control and drive a motor. A L293D chipset which includes 4 dual H-bridges having 0.6A per bridge (1.2A peak) is used to control our 2 bi-directional DC motors and is fed with 8V power source.

Ultrasonic signal based distance sensor is used to avoid obstacles and our HC-SR04 component is capable of providing 2cm - 400cm non-contact measurement function with an operating voltage of 5V.

For the wireless connectivity with the android device, a Bluetooth serial interface is needed to implement the command. Here the android device acts like a master and the Bluetooth transceiver is the slave. BT JY-MCU is the version

of Bluetooth we used with a baud rate of 9600bps and is tolerant at 5V.

All these components were assembled in a hand-crafted chassis with prototyping board and cables; the circuit layout is shown in Fig. 2.

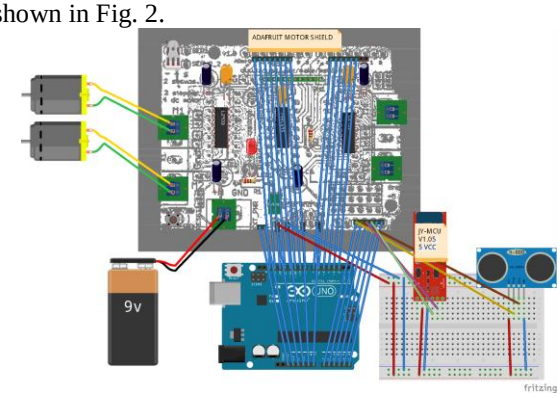


Figure 2. Pin-out sketch of the bot

B. Assembling

Before starting the main programming, we tested each of our components connecting with Arduino with sample sketches to verify everything works right. For the mobility of the robot, a device driver which is simply an abstraction for the intended device is programmed.

We also need a firmware for the distance sensor driver which is actually nothing more than a class that implements the generic interface that is defined for the devices of this class. We used an open source library called arduino-new-ping which works perfectly with our device module.

As a first integration we designed an obstacle avoidance bot so that it can avoid the object and make its own path. Loading our sketch, we encountered unusual responses from the bot and had to deal with this through debugging this embedded system. Checking consequently the logging data through the serial port, at one instance the distance recorded were: 86 cm, 73 cm, and 7 cm. Presumably the 63 and 74cm numbers were about right for the distance to the wall, but the 5cm came out of the blue, without nothing being that close to the bot all of a sudden.

The failures/faulty readings came possibly because of the ultrasonic sensor which estimates the maximum distance ahead from the bounced wave and this process may make errors for handful of reasons. To address our problem, we used averaging to improve the accuracy and not to react to a single reading. To ensure the optimal performance of our bot with minimum amount of delay possible to make decision, we implemented the moving average algorithm. This algorithm works incrementally. We decided to use an average of three readings. Each time we needed to obtain data from the sensor, we take a single reading but don't use it. Instead, we compute and average between that reading and the previous two. The next time and new sensor reading comes, we drop the oldest of the last three readings and average the last two with the new one.

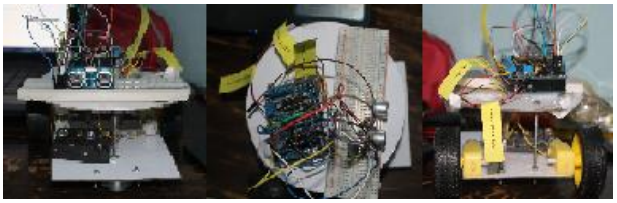


Figure 3. Front, Top and Rear view of the bot respectively

C. Control Mechanism

We wanted the android to act as a wireless source for remote command and hence we needed to design a driver that can abstract the main sketch from having to deal directly with the hardware. In this case, we have two separate pieces to abstract- the most important being the communication method, where our bit use Bluetooth mechanism as a medium. The next is the remote protocol that interprets command codes to implement the actions. Our android controller comes with five directional commands that accepts characters and comes with two sliders, each controlling one side of the vehicle. This directly translates into speeds for the motors we used, so the application can simply send the data from the remote control into the motors.

Our communication protocol driver is embedded inside the communication channel (Bluetooth) acting as an interface to read remote commands. Thus it enables the bot to always react to our native android application.

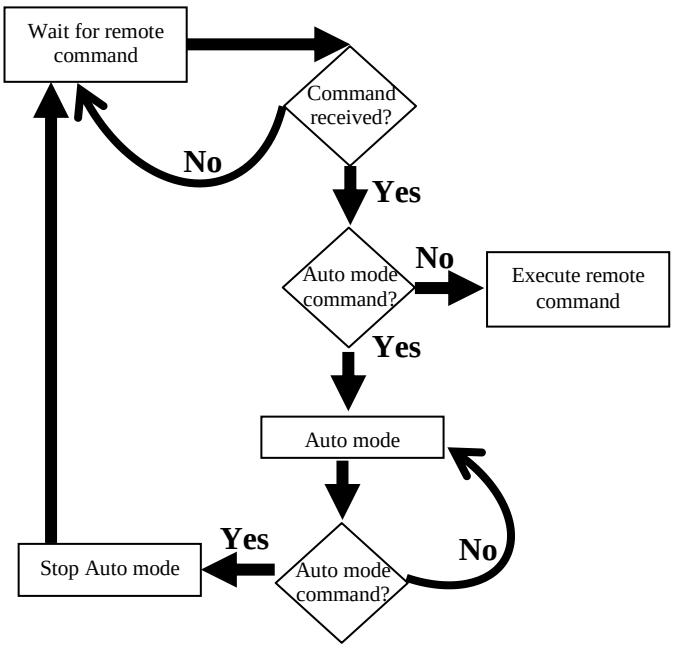


Figure 4. Work-flow for remote command

D. Tracking

Real-time detection and tracking on android platform is achieved through the building blocks of the open-source, cross-platform library called OpenCV. There are four major steps to be incrementally implemented to achieve our target: finding features in the reference image and scene, finding descriptors for each set of features, finding matches between the two sets of descriptors and finding the homography between a reference image and a matching image in the scene.

Putting through each of these features, we used FREAK as DescriptorExtractor, FAST as FeatureDetector and BRUTEFORCE_HAMMING as DescriptorMatcher. We have chosen this combination since it is relatively fast and robust; it

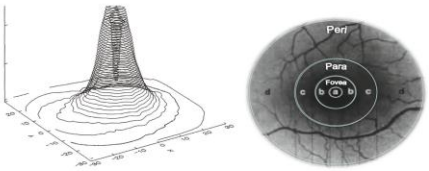
is scale-invariant and rotation-invariant, meaning that the target can be tracked from various distances and perspectives.

Fast Retina Keypoint (FREAK) defines the visual correspondence, object matching relying on representing images with sparse number of keypoints. A cascade of binary strings is computed by efficiently comparing pairs of image intensities over a retinal sampling pattern. The binary descriptor has been constructed by thresholding the difference between the pairs of receptive fields with their corresponding Gaussian kernel,

$$T(P_a) = \begin{cases} 1 \; if \; \left(I(P_a^{T_1}) - I(P_a^{T_2})\right) > 0 \\ 0 \qquad otherwise \end{cases} \quad (1)$$

where $I(P_a^{T_1})$ is the smoothed intensity of the first receptive field of the pair P_a.

Many sampling grids are possible to compare pairs of pixel intensities. In this case, the retinal sampling grid is circular with the difference of having higher density op points near the centre. The density of points drops exponentially as can be seen in Fig. 5.



(a) Density of ganglion cells over the retina (b) Retina areas

Figure 5. Illustration of the distribution of ganglion cells over the retina. The density is clustered into four areas: the foveola, fovea, parafoveal and perifoveal. [7]

The binary descriptor F has been constructed by thresholding the difference between the pairs of receptive fields with their corresponding Gaussian kernel. In other words, F is a binary string formed by a sequence of one-bit Difference of Gaussians (DoG):

$$F = \sum_{0 \le a < N} 2^a T\,(P_a) \quad (2)$$

where P_a is a pair of receptive fields and N is the desired size of the descriptor.

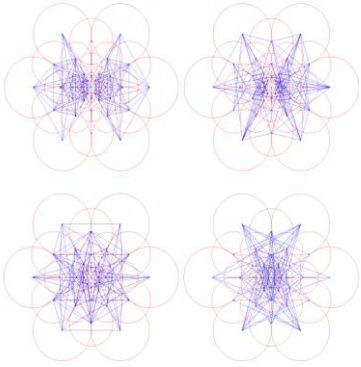


Figure 6. Illustration of the coarse-to-fine analysis. The first cluster involves mainly perifoveal receptive fields and the last ones fovea [7]

Features from Accelerated Segment Test (FAST) is the algorithm we used for identifying interest points in an image.

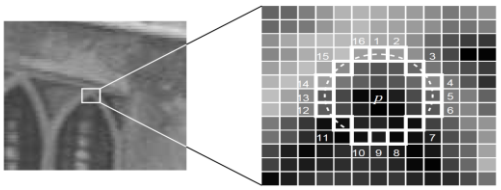


Figure 7. Image showing the interest point under test and the 16 pixel on the circle. [8]

An interest point in an image is a pixel which has a well-defined position and can be robustly detected. The reason behind the use of the FAST algorithm was to develop an interest point detector for use in real time frame rate application on a mobile robot, which have limited computational resources [8].

Based on these algorithms, our application was incrementally developed and its core components include:

1) Video Processing Library and Frame Processor: This is an Android library for processing frames from the device's camera and passing the frame data to the frame processor. The frame processor attempts to detect where the object is located in the frame and then invokes a method in the android application, passing the information obtained from processing the camera frame.

2) Android Application & Robot Controller Library: The application is responsible for initializing the video processing library with a particular frame processor. The robot controller library along with the Arduino sketch provides the object oriented interface for controlling the robot having received command from the Android application.

3) Tracking: This method accepts a rectangle as an argument and is called for each frame. The rectangle represents the co-ordinates of the object being tracked in the processed frame. It works by analysing the rectangle passed in and comparing it to the centre of the frame and to an original object rectangle. By comparing the rectangle to the original object rectangle, the robot can determine if the object is getting bigger or smaller, (i.e. if the objects is moving closer or further away).

E. *Results and Analysis*

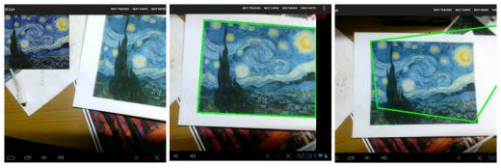


Figure 8. Sequence of tracking frames

This tracking method was implemented in an android device (single core 1GHz processor, 1GB DDR3 RAM) with a robot. The robot can track almost perfectly if the objects move slowly, but it loses the tracked object sometimes if it moves fast. The reason is that this algorithm demands more powerful processor to do real-time processing. Figure 3 shows a sequence of frames of the tracker bot.

III. Tracking Through Pc

Tracking application indigenous to the android platform is what we planned to achieve which would be passing remote

commands wirelessly to the arduino for the bot to act. However, we've faced certain affiliated straight-out constraints in establishing connection with the arduino through wireless communication protocol to the android tracker application.

Besides, real-time tracking with android platform is too a tedious task because of the limitations of the mobile's processor/processing power. The tracker often fail due to continuous appearance changes, distractive objects, or background clutter because of the mobile's lag in processing real-time video. By contrast, many tracking applications are offline since all video frames are available in advance.

Our goal in this part is to perform automatic detection and motion-based tracking of moving objects in a video from a stationary camera. The problems associated with motion-based object tracking can be divided into two parts: detecting moving objects in each frame and associating the detections corresponding to the same objects over time. While our main bot was on the go, we recorded its movement for local data sample, which was then processed in the computer for tracked results.

A. *Methodology*

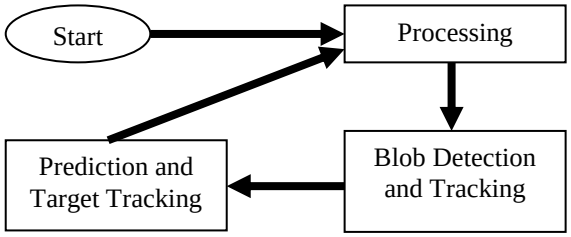


Figure 9. Algorithm flowchart

The detection of moving objects uses a background subtraction algorithm based on Gaussian mixture models. Morphological operations applied to the resulting foreground mask to eliminate noise. Finally, blob analysis [5] detects groups of connected pixels, which are likely to correspond to moving objects.

The association of detections to the same objects is based solely on motion. The motion of each track is estimated by a Kalman filter [6]. The filter is used to predict the track's location in each frame, and determine the likelihood of each detection being assigned to each track.

Track maintenance becomes an important aspect of this task. In any given frame, some detections may be assigned to tracks, while other detections and tracks may remain unassigned. The assigned tracks are updated using the corresponding detection. The unassigned tracks are marked invisible. An unassigned detection begins a new track.

Each track keeps count of the number of consecutive frames, where it remains unassigned. If the count exceeds a specified threshold, the tracker assumes that the objects left the field of view and it deletes the track.

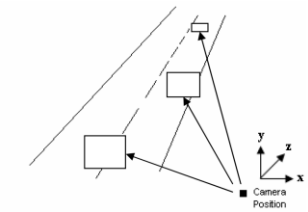


Figure 10. Adaptation of blob sizes. For a fixed camera set-up, blob size of a vehicle is approximated using its distance from the camera position.

B. *Experiment Results and Analysis*

In our application, our approach seemed to perform better which is mainly due to its ability to manipulate the variation to ease the segmentation of the foreground from the background. Moreover, this is based on the minimum reconstruction error over the data needed to be estimated which contribute for the better segmentation.

The tracking experimental results were arranged to track a moving object. The tracking algorithm performed by extracting the moving object from background in each video frame, and then we predict the next position candidates in the next frame. Sequentially the tracking results were based on the routine loop of detecting the objects by applying the data association and the add new hypotheses in Karman filter algorithm respectively and finally we update the frame using the Kalman update function which in turn provides new input in the loop.

Figure 11. Tracking of moving objects [11]

IV. Conclusions

We have constructed a two-wheel Arduino tracker bot based on the concepts of Machine-to-machine (M2M) system. To extend the functionality of our task, we have also implemented offline tracking. Design improvement could be carried for a practical shape so that it serves real-life purpose in surveillance. Wireless onshore connectivity between the Arduino and the Android device could not be achieved because of time complexity; hence this should be addressed.

Acknowledgement

This work is jointly supported by Independent University, Bangladesh and University Grants Commission of Bangladesh under HEQEP Number: CP-3359.

References

[1] Alan J. Lipton, Hironobu Fujiyoshi, and Raju S. Patil. "Moving Target Classification and Tracking from Real-time Video". Proc. IEEE Workshop Application of Computer Vision. P. 8 – 14. 1998

[2] Ismail Haritaoglu, David Harwood, and Larry S. Davis. "W4: Real-Time Surveillance of People and Their Activities". IEEE Transactions on Pattern Analysis and Machine Intelligence. p. 809-830

[3] L. Wixson. "Detecting Salient Motion by Accumulating Directionally-Consistent Flow". IEEE Transactions on Pattern Analysis and Machine Intelligence. p. 774-780. April 2000.
[4] M. Han, W. XU, H. Tao, and Y. H. Gong. An Algorithm for multiple object trajectory tracking. 2004.
[5] A. M. Buchanan and A. W. Fitzgibbon. Interactive feature tracking using k-d trees and dynamic programming. 2006.
[6] C. Stauffer, W.E.L. Grimson. Adaptive Background Mixture Models for Real-Time Tracking. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99), vol. 2 pp. 2246, 1999.
[7] J. Sun, W. Zhang, X. Tang, and H.-Y. Shum. Bi-directional tracking using trajectory segment analysis. 2005.
[8] L. Davis, V.Philomin and R. Duraiswami, "Tracking Humans from a Moving Platform", in Proceedings of the International Conference on Pattern Recognition, vol. 4, p. 4171, 2000, IEEE Computer Society.
[9] R. Gonzalez and R. Woods, "Digital Image processing", 3rd, 2008, Prentice Hall.
[10] R. Abiyev, D. Ibrahim, B. Erin, "Navigation of mobile robots in the presence of obstacles", Near Easr university, Department of Computer Engineering, Mersin 10, Turkey.
[11] J.F. Engelberger, "Health-care Robotics Goes Commercial: The Helpmate Experience" in Robotics, vol.11, pp. 517-523, 1993.
[12] Edward Rosten, Reid Porter and Tom Drummon, "FASTER and better: A machine learning approach to corner detection" in IEEE Trans. Pattern Analysis and Machine Intelligence, 2010, vol. 32, p. 105-119.

About Authors:

Aninda Saha has graduated from North South University in 2014. His research lies in an interesting overlap of Machine Learning and Robotics. He is currently working in operations, research, and development section at Excel BD. He is currently working as a research assistant under the supervision of Dr. M. Ashraful Amin, PhD at Computer Vision and Cybernetics Research Group.

Shuvo Kumar Paul works in the area of Machine Learning and Computer Vision. He received his Bachelor of Science Degree in Computer Science and Engineering from North South University. He has worked on Image processing, Data mining and Machine learning. He is currently working as a research assistant under the supervision of Dr. M. Ashraful Amin, PhD at Computer Vision and Cybernetics Research Group.

Dr. Mahady Hasan is the Head of the Department of Computer Science & Engineering at Independent University, Bangladesh. His research area includes spatial and temporal databases, data mining, robotics and software engineering. He received his PhD from University of New South Wales, Australia.

Dr. Amin is an active contributor to the field of machine learning. He is an editor of Journal of Convergence Information Technology, Journal of Advanced Institute of Convergence IT. He is also a reviewer of *IEEE Transactions on Medical Imaging*, *IEEE Transactions on Fuzzy Systems*, and *International Journal of Imaging and Graphics*, to name a few. Dr. Amin received his PhD from the City University of Hong Kong. He leads the *Computer Vision and Cybernetics* group, where he recruits young and promising research students.