Minor On

**Real Time Object Tracking**



Under the guidance of

**Dr. Malti Bansal**

Assistant professor,  
Department of Electronics and Communication Engineering,

Delhi Technological University

**Submitted by-**

Anadi Anant Jain 2K12/EC/024

Ankush 2K12/EC/029

Bhavyai Gupta 2K12/EC/051

Faizan Ayubi 2K12/EC/065

B.Tech

Electronics and Communication Engineering, Delhi Technological University

CERTIFICATE

This is to certify that the report entitled – “Real Time Object Tracking” is a bona fide record of Minor Project submitted by Bhavyai Gupta (Roll No. 2K12/EC/051) as the record of the work carried out by him under my guidance. It is being accepted in fulfillment of the Minor Project in the department of Electronics and Communication Engineering, Delhi Technological University, Delhi.

Project Guide

Dr. Malti Bansal

Assistant Professor,

Department of Electronics and Communication Engineering,

Delhi Technological University

ACKNOWLEDGEMENT

I express my deepest gratitude towards **Dr. Malti Bansal** , Department of Electronics and Communication Engineering, Delhi Technological University, whose encouragement, guidance and support from initial to final level enabled to develop an understanding of the subject. Her suggestion and ways of summarizing the things made me to go for independent studying and trying my best to get the maximum in the topic, this made my circle of knowledge very vast. I am highly thankful to her for guiding me in this project.

Finally, I take this opportunity to extend my deep appreciation to my family members and my group members **Anadi Anant Jain, Ankush and Faizan Ayubi** for all that they meant to me during crucial times of completion of this project.

Date Bhavyai Gupta

2K12/EC/024

B.Tech (ECE)

INDEX

|  |  |  |
| --- | --- | --- |
| S No | Title | Page No |
| 01 | Abstract |  |
| 02 | Introduction |  |
| 03 | Object Tracking |  |
| 04 | Real Time |  |
| 05 | Sensor Technologies |  |
| 06 | Applications |  |
| 07 | Hardware Specifications |  |
| 08 | Working of Design |  |
| 09 | Code |  |
| 10 | Conclusion |  |
| 11 | Possible Extensions |  |
| 12 | Bibliography |  |

ABSTRACT

The objective of this project is to track any object in real time, meaning we will be completely aware of precise location of object to see its movement in real time.

Here, Bluetooth Module is used for sending data received by Arduino board to global server. With the help of Light sensor, we can detect the motion of object. The proposed system uses Arduino Uno board to receive and analyze data. With the help of global servers, we can save data and act as a server. Motion sensor is also used for detecting the motion of the object.

We will be able to track the number of people in library real-time and head librarian was able to see results on their computer.

INTRODUCTION

The use of video is becoming prevalent in many applications such as monitoring of traffic, detection of pedestrians, identification of anomalous behavior in a parking lot or near an ATM, etc. While a single image provides a snapshot of a scene, the different frames of a video taken over time represents the dynamics in the scene, making it possible to capture motion in the sequence.

Tracking is the problem of generating an inference about the motion of an object given a sequence of images. Good solutions to this problem can be applied to many applications. For example, if we can track a moving person accurately, then we can make an accurate record of his motion. Once we have this record, we can use it to drive a rendering process, i.e. we can modify the motion record to obtain slightly different motions. This means that a single performer can produce sequences he would not want to do in person.

There are three key steps in video analysis: detection of interesting moving objects, tracking of such objects from frame to frame, and analysis of object tracks to recognize their behavior.

In its simplest form, tracking can be defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene. In other words, a tracker assigns consistent labels to the tracked objects in different frames of a video. Additionally, depending on the tracking domain, a tracker can also provide object-centric information, such as orientation, area, or shape of an object.

Tracking objects can be complex due to –

* loss of information caused by projection of the 3D world on a 2D image,
* noise in images,
* complex object motion,
* non rigid or articulated nature of objects,
* partial and full object occlusions,
* complex object shapes,
* scene illumination changes, and
* Real-time processing requirements.

The goal of this project is to group tracking methods into categories and provides comprehensive descriptions of representative methods in each category. We aspire to find solution for this, which require a tracker for a certain application, the ability to select the most suitable tracking algorithm for their particular needs. Moreover, we aim to identify new trends and ideas in the tracking community and hope to provide insight for the development of new tracking methods.

WHAT IS REAL TIME?

*Real time* is a level of computer responsiveness that a user senses as sufficiently immediate or that enables the computer to keep up with some external process (for example, to present visualizations of the weather as it constantly changes). *Real-time* is an adjective pertaining to computers or processes that operate in real time. Real time describes a human rather than a machine sense of time.

In the days when [mainframe](http://searchdatacenter.techtarget.com/definition/mainframe) [batch](http://searchdatacenter.techtarget.com/definition/batch) computers were predominant, an expression for a mainframe that interacted immediately with users working from connected terminals was *online in real time*.

Sensor technologies

There are several motion detection technologies in wide use today:

* **Passive infrared (PIR)**

Passive infrared sensors are sensitive to a person's skin temperature through emitted [black body radiation](http://en.wikipedia.org/wiki/Black_body_radiation) at [mid-infrared](http://en.wikipedia.org/wiki/Infrared#ISO_20473_scheme) wavelengths, in contrast to background objects at room temperature. No energy is emitted from the sensor, thus the name "passive infrared" (PIR). This distinguishes it from the [electric eye](http://en.wikipedia.org/wiki/Electric_eye) for instance (not usually considered a "motion detector"), in which the crossing of a person or vehicle interrupts a visible or infrared beam.

* **Microwave**

These detect motion through the principle of Doppler [radar](http://en.wikipedia.org/wiki/Radar), and are similar to a [radar speed gun](http://en.wikipedia.org/wiki/Radar_speed_gun). A [continuous wave](http://en.wikipedia.org/wiki/Continuous_wave) of microwave radiation is emitted, and phase shifts in the reflected microwaves due to motion of an object toward (or away from) the receiver result in a heterodyne signal at low [audio frequencies](http://en.wikipedia.org/wiki/Audio_frequencies).

* **Ultrasonic**

An ultrasonic wave (sound at a frequency higher than a human can hear) is emitted and reflections from nearby objects are received.Exactly as in Doppler radar, heterodyne detection of the received field indicates motion. The detected [Doppler shift](http://en.wikipedia.org/wiki/Doppler_shift) is also at low audio frequencies (for walking speeds) since the ultrasonic [wavelength](http://en.wikipedia.org/wiki/Wavelength) of around a centimeter is similar to the wavelengths used in microwave motion detectors.

* **Tomographic motion detector**

Tomographic motion detection systems sense disturbances to radio waves as they pass from node to node of a mesh network. They have the ability to detect over complete areas because they can sense through walls and obstructions.

* **Video camera software**

With the proliferation of inexpensive [digital cameras](http://en.wikipedia.org/wiki/Digital_camera) capable of shooting video, it is possible to use the output of such a camera to detect motion in its field of view using [software](http://en.wikipedia.org/wiki/Software). This solution is particularly attractive when the intention was to record video triggered by motion detection, as no hardware beyond the camera and computer is required. Since the observed field may be normally illuminated, this may be considered another *passive* technology. However it can also be used in conjunction with [near-infrared](http://en.wikipedia.org/wiki/Near-infrared) illumination to detect motion in the "dark" (that is, with the illumination at a wavelength not detected by the human eye).

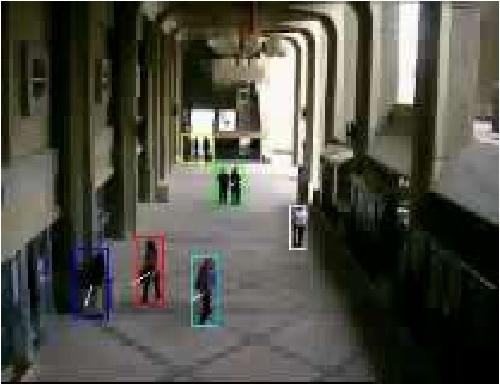
APPLICATIONS OF

REALTIME OBJECT TRACKING

* Traffic Information



* Survilience



* Mobile Robot



USE OF OBJECT TRACKING

* Motion-based recognition, that is, human identification based on gait, automatic object detection, etc.
* Automated surveillance that is, monitoring a scene to detect suspicious activities or unlikely events.
* Video indexing, that is, automatic annotation and retrieval of the videos in multimedia databases
* Human-computer interaction, that is, gesture recognition, eye gaze tracking for data input to computers, etc.
* Traffic monitoring, that is, real-time gathering of traffic statistics to direct traffic flow. Vehicle navigation, that is, video-based path planning and obstacle avoidance capabilities.

HARDWARE SPECIFICATIONS

Arduino Uno

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.



The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Revision 3 (R3) of the board has the following features –

* pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino Due that operates with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
* Stronger RESET circuit.
* Atmega 16U2 replace the 8U2

|  |  |
| --- | --- |
| **Summary of Specifications –** |  |
|  |  |
| Microcontroller | ATmega328 |
|  |  |
| Operating Voltage | 5V |
|  |  |
| Input Voltage (recommended) | 7 – 12V |
|  |  |
| Input Voltage (limits) | 6 – 20V |
|  |  |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
|  |  |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 40mA |
| DC Current for 3.3V Pin | 50mA |
| Flash Memory | 32kB of which 0.5 KB used by boot loader |
| SRAM | 2 KB (ATmega328) |
| EEPROM | 1 KB (ATmega328) |
| Clock Speed | 16 MHz |

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kΩ.

The Arduino Uno can be programmed with the Arduino software.

**Motion Sensor**

A motion detector (motion sensor) is a device that detects moving objects, particularly people. A motion detector is often integrated as a component of a system that automatically performs a task or alerts a user of motion in an area. Motion detectors form a vital component of security, [automated lighting control](http://en.wikipedia.org/wiki/Lighting_control_system), home control, energy efficiency, and other useful systems.



Motion detectors have found wide use in domestic and commercial applications. One common application is activation of automatic door openers in businesses and public buildings. Motion sensors are also widely used in lieu of a true [occupancy sensor](http://en.wikipedia.org/wiki/Occupancy_sensor) in activating street lights or indoor lights in walkways (such as lobbies and staircases). A motion detector may be among the sensors of a [burglar alarm](http://en.wikipedia.org/wiki/Burglar_alarm) that is used to alert the home owner or security service when it detects the motion of a possible intruder. Such a detector may also trigger a [security camera](http://en.wikipedia.org/wiki/Security_camera) in order to record the possible intrusion.

**Door Sensor**



**Light Sensor**

The light sensor measures how much light is shining on it. It has two modes: “light” and “dark.” In “light” mode, the more light shines on the sensor, the higher the signal it sends out. In “dark” mode, it's just the opposite – the signal increases the darker it gets. You can use a screwdriver to adjust the sensitivity. Use a bargraph to see how it's working!



WORKING OF THIS DESIGN

Step 1 Arduino Necessities

* 1 Arduino Uno Board
* 1 PIR (Passive Infrared) Sensor
* 3 M-F Jumper Wires
* 1 Pi Cobbler
* 1 Breadboard
* 1 Bluetooth Module

Step 2 Getting it set up

* Download the Arduino IDE and install
* Download the Processing IDE and install

Step 3 - Connecting the wires to the sensor

* Connect 3 of the M-F jumper wires to the GND (Ground), OUT, and +5V 2013-08-22\_2251

Step 4 – Connecting the Parts on the ARDUINO

* Connect the pi cobbler/cable to the ARDUINO and bread board
* Connect the Black wire coming from Ground on the PIR sensor to the Ground on the breadboard
* Connect the red wire coming from the 5v on the PIR sensor to the 5v on the breadboard
* Connect the yellow wire coming from the OUT on the PIR sensor to port 8 on the breadboard

Step 5 – Adding the Code

* Arduino
* Processing

Step 6 – Running the Program

* Then open up Processing IDE and see Output Window

Step 7 – Check Real Time Reports in Global Server

* Now look in the Real Time events to see the actions come through.

CODE

**Arduino**

int sensor = 8;

void setup() {

pinMode(sensor, INPUT);

Serial.begin(9600); //Begin Serial Communication

}

void loop() {

int sensorState = digitalRead(sensor);

Serial.println(sensorState); //Send IR Sensor value to computer

delay(100);

}

Processing

import simpleML.\*;

import processing.serial.\*; /\* Needed for Serial Communication \*/

// A Request object, from the library

HTMLRequest htmlRequestSend;

HTMLRequest htmlRequestReceive;

// the URL for the php\_cell

String url = "http://cloudstuffs.com/arduino/data.php";

// Data received from the serial port

int val;

boolean receiving = false;

boolean useSerial = false;

/\* Global variables \*/

Serial comPort;

String [] comPortList;

String comPortString;

/\*--------------------------------------------------\*/

void setup(){

size(100,100); /\* Set the size of the window \*/

background(0); /\* Set the background to black \*/

/\* Get the available com ports. If there is at least one

com port available, then start communicating on it.

If there are more than one com ports available, we will

only open the first one (i.e. comPortList[0])

The bufferUntil('\n'); statement will generate a serial Event

when it reads a carriage return \*/

comPortList = Serial.list();

if(comPortList.length>0){

comPort = new Serial(this, comPortList[0], 9600);

comPort.bufferUntil('\n');

}

}

/\*--------------------------------------------------\*/

void draw(){

/\* The serialEvent function will update the display \*/

if ( receiving && myPort.available() > 0 && useSerial ) { // If data is available,

val = myPort.read(); // read it and store it in val

background(val); // Set background to white

//println(val);

// Create and make an asynchronous request

// PUT THE RIGHT GROUP NUMBER HERE

htmlRequestSend = new HTMLRequest( this, url + "?r=" + val );

// <div id='$count'><div id='key-$count'>f</div><div id='val-$count'>45</div></div>

htmlRequestSend.makeRequest();

}

if ( !receiving ) {

htmlRequestReceive = new HTMLRequest( this, url );

htmlRequestReceive.makeRequest();

}

}

/\*--------------------------------------------------\*/

void serialEvent(Serial cPort){

comPortString = cPort.readStringUntil('\n');

if(comPortString != null) {

comPortString=trim(comPortString);

/\* Print to the Debug screen in Processing IDE \*/

println(comPortString);

}

}

void netEvent(HTMLRequest ml) {

if ( !receiving ) {

String html = ml.readRawSource();

parseHtml( html );

}

}

void parseHtml ( String html ) {

// print the values on the website

int countData = 0;

String startStr = "<div id='key-"+countData+"'>";

int start = html.indexOf( startStr );

while ( start >= 0 ) { // start == -1 means there is no data to be parsed

start += startStr.length();

String tempStr = html.substring( start, html.length() );

int end = tempStr.indexOf( "</div>" );

String theKey = html.substring( start, start + end );

print( theKey + " - " );

startStr = "<div id='val-"+countData+"'>";

start = html.indexOf( startStr );

start += startStr.length();

tempStr = html.substring( start, html.length() );

end = tempStr.indexOf( "</div>" );

String theValue = html.substring( start, start + end );

­­­­­

println( theValue );

// send the values back over the serial port

if ( useSerial )

if ( int(theValue) <= 255 ) myPort.write( int(theValue) );

countData++;

startStr = "<div id='key-"+countData+"'>";

start = html.indexOf( startStr );

}

}

CONCLUSIONS

In this Project, we present an extensive survey of object tracking methods and also give a brief review of related topics. We divide the tracking methods into three categories based on the use of object representations, namely, methods establishing point correspondence, methods using primitive geometric models, and methods using contour evolution. Note that all these classes require object detection at some point. For instance, the point trackers require detection in every frame, whereas geometric region or contours-based trackers require detection only when the object first appears in the scene. Recognizing the importance of object detection for tracking systems, we include a short discussion on popular object detection methods. We provide detailed summaries of object trackers, including discussion on the object representations, motion models, and the parameter estimation schemes employed by the tracking algorithms. Moreover, we describe the context of use, degree of applicability, evaluation criteria, and qualitative comparisons of the tracking algorithms. We believe that, this article, the first survey on object tracking with a rich bibliography content, can give valuable insight into this important research topic and encourage new research.

POSSIBLE EXTENSIONs

* Fine tuning of the various threshold parameters to get a better and more sensitive real-time object tracking system.
* On the basis of the system, build a new tracking system that can still work if the illumination of the reference background changes.
* Enhance this system to make it capable of tracking the actual shape of the entering object.
* Modify this system to make it capable of tracking more than two objects at the same time.
* Possible Home Server

BIBLIOGRAPHY

* D.-S. Jang, G.-Y. Kim, and H.-I. Choi, “Model-based tracking of moving object,” Pattern Recognition, Vol. 30, pp.999-1008, 1997.
* J. Kim and J. Woods, “3-D Kalman filter for image motion estimation,” IEEE Transactions on Image Processing. Vol. 7, No. 1, pp. 42-52, Jan. 1998.
* S. Lee and Y. Kay, “An accurate estimation of 3-D position and orientation of a moving object for robot stereo vision: Kalman Filter Approach,” IEEE International Conference on Robotics and Automation, Proceedings., Vol.1, pp. 414 - 419, May 1990.
* [www.google.com](http://www.google.com)
* [www.youtube.com](http://www.youtube.com)
* [www.wikipedia.com](http://www.wikipedia.com)
* AGGARWAL, J. K. AND CAI, Q. 1999. Human motion analysis: A review. Comput. Vision Image Understand. 73, 3, 428–440.
* ALI, A. AND AGGARWAL, J. 2001. Segmentation and recognition of continuous human activity. In IEEE Workshop on Detection and Recognition of Events in Video. 28–35.
* AVIDAN, S. 2001. Support vector tracking. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 184–191.
* BADDELEY, A. 1992. Errors in binary images and an l version of the haus- dorff metric. Nieuw Archief voor Wiskunde 10, 157–183.
* BALLARD, D. AND BROWN, C. 1982. Computer Vision. Prentice-Hall.
* BAR-SHALOM, Y. AND FOREMAN, T. 1988. Tracking and Data Association. Academic Press Inc.
* BARRON, J.,FLEET, D., AND BEAUCHEMIN, S. 1994. Performance of optical flow techniques. Int. J. Comput. Vision 12, 43–77.
* BEAULIEU, J. AND GOLDBERG, M. 1989. Hierarchy in picture image segmentation: A step wise optimization approach. IEEE Trans. Patt. Analy. Mach. Intell. 11, 150–163.
* BERTALMIO, M., SAPIRO, G., AND RANDALL, G. 2000. Morphing active contours. IEEE Trans. Patt. Analy. Mach. Intell. 22, 7, 733–737.
* BEYMER, D. AND KONOLIGE, K. 1999. Real-time tracking of multiple people using continuous detection. In IEEE International Conference on Computer Vision (ICCV) Frame-Rate Workshop..
* BIRCHFIELD, S. 1998. Elliptical head tracking using intensity gradients and color histograms. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 232–237.
* BLACK, M. AND ANANDAN, P. 1996. The robust estimation of multiple motions: Parametric and piecewisesmooth flow fields. Comput. Vision Image Understand. 63, 1, 75–104.
* BLACK, M. AND JEPSON, A. 1998. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. Int. J. Comput. Vision 26, 1, 63–84.
* BLAKE, A. AND ISARD, M. 2000. Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion.
* Springer. BLUM, A. AND MITCHELL, T. 1998. Combining labeled and unlabeled data with co-training. In 11th Annual Conference on Computational Learning Theory. 92–100.
* BLUM, A. L. AND LANGLEY, P. 1997. Selection of relevant features and examples in machine learning. Artific. Intell. 97, 1-2, 245–271.
* BOSER, B., GUYON, I. M., AND VAPNIK, V. 1992. A training algorithm for optimal margin classifiers. In ACM Workshop on Conference on Computational Learning Theory (COLT). 142–152.