IoTracker: Home-made Tracking System using Core-sets and the Internet of (Tracking) Things

Soliman Nasser Ibrahim Jubran Dan Feldman
The IoT and Big Data lab,
University of Haifa

ABSTRACT

We present a low-cost motion tracking system that is based on regular web-cameras that are connected to single-board mini-computers, usually used for the "Internet of Things". To compensate the relatively weak hardware, we provide approximation algorithms for the PnP tracking problem, with provable guarantees, based on a data reduction technique known as core-sets.

We show applications of our system for (i) People navigation inside buildings such as homes and malls, using guiding autonomous safe and low-cost drones, (ii) Autonomous low-cost and programmable toy robots, including helicopters and Lego cars and , (iii) Plug-ins for using our tracking system with the classic kids' programming language "Scratch".

Existing motion capture systems for such applications (e.g. Optitrack and Vicon) use dedicated hardware and workstations that cost thousands of dollars, and thus exist mainly in research labs. Using the open-source algorithms and instructions, our tracking system can be built by non-expert at home ("Do it yourself") at a cost of few dozens of dollars, and can operate without a laptop or a desktop computer. This makes the system affordable for massive use in smart-homes, cities, malls and hospitals as well as in developing countries.

1. MOTION CAPTURE SYSTEMS

The aim of a motion capture system is to track a given object, usually a robot or a human, and log its coordinates in real-time. Using such a system we can control, for example, an autonomous robot or drone that does not have any sensors on board except a receiver, by a computer that is connected to its remote transmitter. A popular motion capture system consists of three to few dozens of cameras that are spread around the room. Each camera is mounted on the wall and calibrated before its first usage. Each desired object in the room is registered to the system before the experiment, as explained below. During the experiment each camera locates the desired object in its frame and sends the result to a server in the room that collects the information from the cameras in real-time and estimates the location of each object in the room. The location of each registered object is then sent to the client applications of the user, in a rate of at least few dozens of frames per second.

Figure 1: (left) 3 IoTracker clients installed near the ceiling of our CS department. (right) The RC ladybird $30 quad-copter that we turned into IoTracker-based autonomous robot.

IR cameras. The most accurate motion capture systems in the market are based on cameras with infra-red filters and infra-red lights. In this case, a few reflective markers (usually small balls) must be attached to each object in the scene and registered in the system.

The main computation problem is the actual location of the object based on the 2D coordinates of the markers in the image. See the PnP problem below.

RGB cameras. In the case of RGB cameras, there are usually no markers on the object and computer vision algorithms are used to detect features and edges of the object, e.g. using SIFT and SURF algorithms.

Existing Systems. Probably the most two popular tracking systems in the market are Vicon and Optitrack, both of them are based on IR cameras. Each Optitrack camera costs $600 to $6000 according to the web-site. The quality is from 0.3MP resolution, 100 Frames Per second, and 46 degrees for the field of view, to 4.1MP, 240 FPS, and 82 degrees. Vicon cameras are considered more expensive. These systems are based on USB or GigE cables between the clients and the server, as well as strong workstation computer that is used as

the server. RGB trackers are usually suggested as software for a single web-camera and a laptop or desktop computer, and not as a tracking system, e.g. CMT or openTLD.

2. IOTRACKER

In this section we present the IoTracker motion capture system that we designed and implemented. We provide code for both IR and RGB detection, by using the appropriate filters on the cameras. The system consists of one server node (without a camera) and client nodes, each connected to a development board as follows. Unlike the existing systems above, the communication between the clients and the server is wireless. In addition, we implemented the server on another low-cost development board, without the need for a workstation computer or even a laptop.

Of course, IoTracker is significantly less accurate and fast than the professional expensive motion tracking systems. Never the less, it is more than enough for all the applications that we implemented and suggest in Section **??**. For example, while our system is probably not suitable for aggressive maneuvers, we hovered a toy quadcopter without any sensors except a receiver with an accuracy of less than 0.5cm.

2.1 Deployment

While most of our code is generic, we suggest to use the following hardware setup.

- Web Camera: SONY Play Station 3 Eye or Microsoft LifeCam ($8–$15). The minimum number of cameras needed for IR tracking is two, and for RGB tracking the minimum is one. For IR tracking, an IR filter should be inserted to the camera. This can be done as explained in existing tutorials on the web.
- Client's board: Intel's Galileo Gen 2, or Odroid U3 ($40–$60). Each camera is connected to its own board using a USB cable. The board should be equipped with a wifi adapter for communication to the server, and batteries or power cord.
- Server's board (same as Camera's board). This is the single board that collects the tracking information from all the other cameras through the communication protocol that we implemented based on UDP.
- IR LEDs ($4-$8). For the IR mode of IoTracker, a ring of infra red LEDs should be put on top of each camera. Such rings that are usually used for CCTV security cameras are available e.g. in Amazon. The LEDS are also connected to the batteries (12V).
- (optional) Arduino Uno R3 and PIR Sensor ($20-$25). For saving energy, we connect each client board or pair of clients to an Arduino Uno R3 controller that receives input from a motion detector (PIR sensor) and turn on/off the connected client when there is no motion for given amount of time.

2.2 Novel Algorithms

Existing motion tracking systems are based on algorithms that run very fast mainly because they run from dedicated embedding devices inside the cameras. The main challenge in designing IoTracker is to enhance the relatively weak hardware by very efficient tracking algorithms that run on the "Internet of Things" boards that support generic code.

In particular, the main problem that we had to solve in each camera on real-time, for both the IR and RGB setting, is the PnP problem, which stands for Perspective-n-Point. This is a problem in computer vision which estimates an object pose from given n 3-D points and their projections in an image.

This problem is known to be NP-hard for growing n, so mainly heuristics are used in practice. For the case of IR markers (when n is relatively small) we provide an algorithm that returns an answer which is provably 2-approximation with respect to the optimal solution, in time $O(n)$.

For the case of RGB (when n is a large number of features in the image), we suggest a core-set for the PnP problem. A core-set is a small carefully-chosen subset C of the input set P, such that running a specific algorithm (in our case, a PnP solver) $A(C)$ on the coreset C , yields a result $A(C) \sim A(D)$ which is a provable approximation to the output of running the same algorithm on P. We provide an algorithm for computing such a coreset that introduces $(1 + \varepsilon)$ multiplicative factor approximation, and whose cardinality is $O(1/\varepsilon)$ input points (markers). We prove that such a coreset exists for every input set P of n points, and can be computed in $O(n)$ time. On this coreset we run our approximation algorithm, but we can also run existing heuristics, and expect similar but faster results as the corresponding runs on the original input.

2.3 Provided Applications

To demonstrate the capabilities and qualities of our system, we implemented the following open-source applications and related systems that we wish to demonstrate and publish during the conference:

- Autonomous low-cost quadcopter. We connected an Arduino device that controls a popular toy quadcopter in order to have an autonomous robots. The input to the system is the current position of the quadcopter, based on IoTracker, and the desired position in each moment. The commands are transmitted to the robot using PWM radio signals.
- Guarding Angel. We implemented an algorithm for tracking people based on RGB images from IoTracker. Together with the system in the previous bullet, we obtained a system that helps people to navigate inside a building by following a quadcopter. The quadcopter gets the target location from the guest based on voice commands, using existing speech recognition system (Sphinx).
- Autonomous Lego Toys. We reverse-engineered the remote IR controllers of Lego,and provide Arduino C code that simulates the remote controllers. Combining this with IoTracker that reports the current position of the Lego toy, we obtained a closed loop for programming Lego autonomous components, such as cars, trains or any component that is connected to a Lego receiver.
- 3-D Programming. "Scratch" is a popular computer language, specially designed for kids. Since it is based on open-source, we were able to add the commands "goto (x,y,z)" that sends a robot to a specific place in the real world, and "(x,y,z)=current-location" which reports the current location of the robot based on IoTracker. This function supports the quadcopter, and Lego IR based toys above.