Low-cost Optical Tracking System using Wii Remote

Wei Zhang, Yi Gong
Department of Computer Science
University of California, Santa Barbara
wei, ygong@cs.ucsb.edu

Abstract

In this project, we proposed a low-cost 3D position tracking system using Wii remote controller and IR markers. Wii is a very popular console for home entertainment. The Wii Remote controller brings revolutionary HCI technologies to real world and inspires many new applications. However, most of them only uses the basic functionalities but not exert Wii's full potential. This work is distinguished from exist tracking systems by some key features. The first is we exploit the multi-view capability of the low-cost Wii remote controller, which contains an IR camera and on-board hardware for signal detection, so that provides the possibility to create further exciting interactive applications with mininum cost. The second is we hide the complexity of computer vision since our users may not be vision professionals. Currently we have built a prototype system and implemented basic multi-view geometry algorithms, the final goal of this project is to provide this as an opensource library for other Wii developers to create their interactive applications.

1. Introduction

Optical tracking system is a well studied area in the past decades, many commercial systems are available in the market. However, most of them are complicated and very expensive, built only for professionals.

The Nentindo Wii console has a built-in IR sensor in its remote controller, with on-board hardware processing capability to track up to four IR markers simultaneously. This give us the possibility to build the cheapest motion capture system ever, and if consider that Wii has already been sold to over 20M families, our system will only cost them only a few IR markers. Such kind of low-cost and tiny motion capture system can be used as a basis to build many new applications for home interactive entertainment.

Several interesting application of Wii remote has been developed and produce huge impact. The most famous ones are the head tracking and multi-touch whiteboard developed by Johnny at CMU[**?**] [**?**]. However, all of current applications only use a single Wii remote for its built-in 2D tracking capability.

In our project we are going to build a 3D tracking system using two Wii remote, this is a classic 3D reconstruction from multiple view problem which has been explored by many researchers. For the 2 view cases, a lot of studies have been made on the fundamental matrix and calibrated or uncalibrated image marching [**?**],[**?**] [**?**] [**?**]. [**?**] is also a good book that cover these issues. In our method, we will calibrate our camera by [**?**]'s algorithm first, then calculate the fundamental matrix and reconstruct 3D refer to [**?**].

In our scenario, user are allowed to place two cameras at anywhere he like, as long as the markers are visible. So at first the system need to calibrate the cameras for their extrinsic and intrinsic parameters. In order to simplify this process we build a calibration board which has 4 markers attached on a square with known positions. During the initialization, the system detect these four points and use the CalibrateCamera2 function in OpenCV to find out necessary parameters about cameras.

After the camera's intrinsic and extrinsic parameters are known, the system is ready to track. There are mainly two problems here. The first one is to find the coorespondency between object points from different views. Once this spatial coorespondency is determined then inhomogenous linear triangulation method can be applied to reconstruct the points in 3D space. The second problem involves matching two sets of 3D points representing detected markers at two consecutive frames, respectively, thus finally provide the trajectory of tracking objects.

The rest of the article is arranged as follows: In section 2 we introduce the design of hardware and software platform that our algorithms are running on, then section 3 provides the details of the camera calibration, temporal and spatial coorespondency algorithms. In section 4 we provides a set of experiment results and discuss the performance of our system and algorithms.

2. Platform

2.1. Wii Remote

Wii remote is Wii's main input device. It contains a 1024x768 infrared camera with built-in hardware on-board processing to provide tracking of up to 4 points (e.g. IR led lights) at 100Hz, and it also contains a +/-3g 8-bit 3-axis accelerometer which also operating at 100Hz.

Wii remote can be connected to the console or PC via bluetooth, this feature gives us the opportunity to explore other applications for this hardware rather than gaming. It uses the standard Bluetooth HID protocol to communicate with the host, which is directly based upon the USB HID standard, so it will appear as a standard input device to any Bluetooth host. However, the Wiimote does not make use of the standard data types and HID descriptor, and only describes its report format length, leaving the actual contents undefined, which makes it useless with standard HID drivers. The Wiimote actually uses a fairly complex set of operations, transmitted through HID Output reports, and returns a number of different data packets through its Input reports, which contain the data from its peripherals.

Wii community has been succeeded in trying to discover the control and communication with the Wii remote, there are several opensource software beed released. In this project, we use our modified driver based the opensource Wiimote library to drive multiple input devices.

2.2. IR Markers and Calibration

The most sensitive wavelength for Wii remote is between 800 and 950 nm. The LEDs that we use are Vishay TSAL6400/6200s running at 100mA.

Our IR markers are very simple and easy to build. Each IR marker is actually an AA battery and an small IR LED connected by a switch.

The calibration board is a plane board with 4 LEDs on it, these LEDs are fixed at the corner of a 1 foot size square on the board, and they can be connected to a battery on the other side.

3. Software Architecture

We want to build our software as an open 3D tracking library so that other developers can use it to make their own applications. In this project, the software structure can be described as following from bottom to top:

- The bottom level is a modified Wiimote lib. This is the hardware driver that enable us to communicate and control multiple Wii remotes through bluetooth wireless, the 'image' we get from Wii remote is actually coordinates of points in camera's image plane. Data are provided through a socket service so that developers can access the Wii remote raw data using any platform and programming languages.
- Camera calibration estimates the intrinsic and extrinsic parameters of two cameras during initialization.
- At the stereo vision level we use the camera parameters and detected marker positions to produce their 3D position.
- Finally the tracking results are provided to interactive applications, for example, to control the role's movement in a fps game base on the player's certain specific postures.

4. 3D Reconstruction Based on Two Views

The 3D reconstruction of our system can be devided into three steps: camera calibration, point pair matching and 3D position calculation.

4.1. Camera Calibration

Although a lot of researchers have exploited the methods of multiple view stereo problem based on uncalibrated cameras, these methods need at least 7 marks[**?**], as fundamental matrix has 7 freedoms. Due to the 4-marks limitation of wii driver, we cannot apply enough marks to implement our system without camera calibration. Therefore, we take use of Zhang's calibration method by calling the routine of OpenCV. At the first stage of our method, we will put a measured square plane, with four infrared LEDs attached around its corners, in front of the two infrared cameras. For each camera, our application will analysis the location of the four 2D coordinates and bind them with correspond predefined 3D coordinates automatically. Then it will do the calibration and get the rotation matrics R_1, R_2 and translation matrix t_1, t_2. To easier our further work of essential matrix calculation and 3D reconstruction, we do a world transformation to make the transformation matrix of the first camera change from $(R1 \mid t1)$ to $(I \mid 0)$. It can be proved that the transformation matrix of the second camera will becomes

$$(R_2 R_1^{-1} \mid t_2 - R_2 R_1^{-1} t_1) \tag{1}$$

after this world transformation.

4.2. Point Pair Matching

In the point pair matching stage, we take use of the epipolar geometry to find the one-to-one relationships between two groups of points from the two camera's view. This relationship is built based on the essential matrix E, which is similar to fundamental matrix but just applied to normalized image coordinates:

$$x'_n \, E \, x_n = 0 \tag{2}$$

Here x_n and x'_n are normalized image coordinates of the same 3D points in view1 an view2. It is obvious that x_n and x'_n can be get by just applying the inverse instrinsic matrics of the cameras to their original coordinates x_o, x'_o:

$$x_n = K_1^{-1} x_o \quad and \quad x'_n = K_2^{-1} x'_o \tag{3}$$

where K_1 and K_2 are the instrinsic matrix of the two cameras. The essentail matrix can be calcualted from the transform vector and rotation matrix between the two cameras:

$$E = [t]_\times R \tag{4}$$

As we've mentioned in last subsection, the rotation matrix and translation vector between our two cameras are:

$$R = R_2 R_1^{-1} \tag{5}$$

$$t = t_2 - R_2 R_1^{-1} t_1 \tag{6}$$

Getting the essential matrix, Equation 2 tell us the property of correspondant point pairs. Actually, recally that

$$l' = E x_n \tag{7}$$

we can see the geometry meaning of Equation 2: x_n 's correspondant point x'_n should lie on the the epipolar line l' in the second view. However, such requirement can only be meet in ideal situation. Noises and errors brought in the whole process make this property unreliable. In such cases, the problem becomes a optimal solution search based on energy minimization. The energy here is defined as a function of distance from the 2D point to its pair's epipolar line:

$$d(x, l')^2 + d(x', l)^2 \tag{8}$$

We try all possible point pairs and keep the one that minimizes the energy.

4.3. 3D Coordinates Calculation

To calculate the 3D position of x and x', we use the inhomogenous linear triangulation method[**?**]. Knowing that $x = PX$ and $x' = P'X$, where P and P' are the projection matrix of camera1 and camera2, we can apply the factor that $x \times (PX) = 0$ and $x' \times (PX) = 0$. Each of them will give us three linear equations with dependency that allows us to elimiate one. Finally, it will give us four linear equations together. If we reorginanize the coefficients and variables, we can get the linear equations in the form of $AX = 0$, where A:

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p^{1T} \\ y'p'^{3T} - p^{2T} \end{bmatrix} \tag{9}$$

p^{iT} here represents the ith row of P and similarly p'^{iT}. Replace the X with $(X, Y, Z, 1)^T$, we can derive this 4×4 homogenous system to a 4×3 inhomogenous one. The right hand side vector b will just be the negative fourth column of original A matrix. This inhomogenous can be solved by many mature factorization method, like SVD or QR decomposition, which will gives the least square solution if b is not in the range space of A. This solution gives us the 3D coordinates of the 2D point pair x and x'

5. Planned Experiments

Our experiment includes two phases. Phase1 would be evaluating our stereo vision algorithm. For the camera calibration, we will use two cameras, with 4 markers on a square board to measure the error of openCV's algorithm. For spatial correspondency algorithm, we are going to use 2 cameras and 1 marker, for different angle and distance, measure the average error and jitter of calculation result and physical measurement. For testing the temporal correspondency, we plan to use 4 markers, stick them to two hands and elbows of a person, and see how well can our algorithm match markers to their corresponding places, important parameters including the distance between the person and cameras, and marker's moving speed.

Phase2 would be building demo applications to show the potential of this system. Base on previous experiment, this first demo would be place markers on the arms to show control of a simple boxing/acting game. If we have enough time, a complicated application could be attach 4 markers to legs and arms respectively, then try to match the person's posture to our pre-defined templates, each template will correspond to a motion(such like jump, run, squat) or a command of the role in the virtual world.

The expected result is that our system should work well when get perfect input – no occlusion and motion is normal speed. We will try to strengthen our application by estimate points' position, and tolerate error of detected points data by march the most likely points when searching correspondence. These kind of experiments of robustness is not predictable yet. But we want it can handle short and partial occlusion and have the ability of error tolerance.