

Using multiple WiiMote cameras to control a game

Eike Dehling, e.e.dehling@student.utwente.nl, s0052663,
Human Media Interaction, University of Twente

ABSTRACT

The Wii Remote controllers can be used to detect the position of IR lights. They can be connected to a workstation using Bluetooth. Two controllers attached to one workstation allow to determine the position of an IR light from two views.

Stereovision algorithms combine the differences in two views of a scene to extract 3D information. Calibration calculates camera parameters, triangulation determines the 3D position of a point using these parameters.

For calibration the RANSAC and LMedS algorithms give the best results, while the normalized 8 point algorithm is not far behind. Triangulation works best with the polynomial algorithm.

Using two Wii Remote controllers and the stereovision algorithms described in this report, the position of an IR light can be estimated.

1 Introduction

The initial plan for the project was to build a ping-pong game controlled using a real bat. IR lights and cameras were selected to detect the 3D position of the bat.

The Nintendo Wii controller has a high performance IR camera with built-in tracking of lights. This IR camera operates at 100hz which is significantly faster than a normal web cam. It has a resolution of 1024x768 pixels. This makes the camera ideal to control games [Lee08]. Because the Nintendo Wii controller interfaces to the game console using the standard Bluetooth protocol, it can also interface with a regular workstation. The high quality of the camera and the option to interface with a workstation make this an ideal tool for the project.

Time constraints meant the actual game was not built. The final research question became: Which stereovision techniques are best suited to determine the 3D position of an IR light using the WiiMotes camera.

In section 2.1 some general bits about stereovision are explained, followed by a more in-depth explanation of the stereovision process in 2.2 and 2.3. Then in 2.4 the WiiMote controller and its peculiarities are covered. A little is explained about the software produced in 2.5. Finally in section 3 the reports ends with the results and some recommendations.

2 Details

This section goes into more detail on the techniques used in interfacing with the WiiMote. It also describes stereovision, algorithms from computer vision which allow to estimate 3D positions of objects.

2.1 Stereopsis

Stereopsis is the process leading to perception of depth. The perception of depth comes from combining the differences in two projections of a single scene. [FP02]

In computer vision, epipolar geometry is used to estimate depth from two images of a scene. There are two parts in this process: determining the epipolar geometry of two cameras (also called calibration) and reconstructing the scene from recorded images (called triangulation). [HZ00] [FP02]

There are two important properties of a camera system that are captured by calibrating it. Each camera projects and distorts images in a different way (Intrinsic camera properties). Also relative position and orientation of the two cameras need to be determined (Extrinsic properties). The intrinsic and extrinsic properties can be captured in the fundamental matrix [HZ00] [LF95]. Using points visible in both images (point matches) one can estimate the fundamental matrix. We will elaborate on different estimation algorithms in section 2.2.

Triangulation uses the fundamental matrix and matching points to reconstruct the 3D position. Since there usually are errors in the measured points, there is no exact solution for the 3D position and it has to be estimated. Several approaches are explained in section 2.3.

An introduction in epipolar geometry is needed to understand the following sections. Epipolar geometry is the geometry of two cameras viewing a single scene. See figure 1 for the model of a scene. There are two cameras C and C' , which project on the two image planes. Both cameras see object M and project it into their image plane as x and x' respectively. The projection of the other camera into the image plane is called the epipole. The epipolar plane is the plane through point M and the cameras C and C' . The two lines where the epipolar plane intersects the image plane are called epipolar lines.

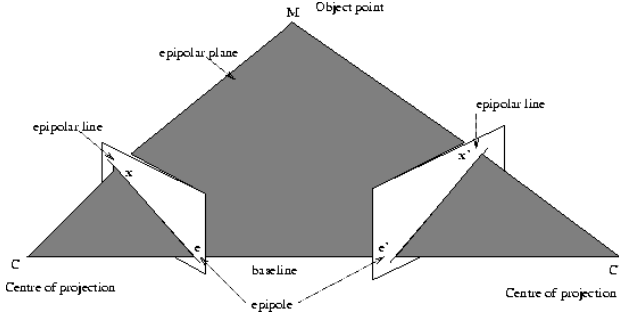


Figure 1: Epipolar geometry (image taken from Robyn Owens website at University of Edinburgh)

Because both projected points x and x' lie on the epipolar plane, the point x visible in one image projects onto the epipolar line in the other image. This is called the epipolar constraint.

2.2 Estimating the fundamental matrix

The fundamental matrix F relates matching points p and p' on the two image planes. It expresses where a point p visible in one image plane can be on the other image plane: on the epipolar line in the other image plane. One epipolar line is defined by the following equation:

$$F.p = 0 \quad (1)$$

Because the point p' also lies on that epipolar line the following equation holds:

$$p'^T . F . p = 0 \quad (2)$$

Points p and p' can be written as $p = (x, y, 1)$ and $p' = (x', y', 1)$. If the components of F are then written as f_{nm} we can write this equation in another form:

$$x'.x.f_{11} + x'.y.f_{12} + x'.f_{13} + y'.x.f_{21} + y'.y.f_{22} + y'.f_{23} + x.f_{31} + y.f_{32} + f_{33} = 0 \quad (3)$$

Now define f as a vector consisting of the rows of F concatenated. The previous equation can then be written as follows:

$$(x'.x, x'.y, x', y'.x, y'.y, y', x, y, 1).f = 0 \quad (4)$$

From n points that are visible in both images, one gets n linear equations in f . The last form of that equation is convenient because of the way matrix multiplications work: The vector parts can be used as rows in a matrix and the equations turn into one linear equation. Usually there is no exact solution to this system of equations and an estimate needs to be found.

There are several algorithms that can produce such an estimate. The problem in finding an estimate are

measurement errors, and the ways to deal with these incorrect points. The algorithms below are described and their accuracy analyzed in more detail in [Zha98].

- **7 Point Algorithm** The minimum number of matching points required to estimate a fundamental matrix using equation 4 is 7. (See [HZ00], [FP02]) This algorithm uses that minimum number of points to determine the fundamental matrix. It does not compensate for measurement errors.
- **8 Point Algorithm** This algorithm works with 8 or more matching points. It determines the least squares solution of the system of equations to minimize the error in the fundamental matrix. Though because of the way the algorithm works, measurement errors still deteriorate the results. This algorithm is analyzed in detail in [Har97]. This paper includes improvements to the basic 8 point algorithm as well.
- **RANSAC, LMedS** Using 8 or more matching points, the fundamental matrix is determined for subsets of matching points (using the 7 point or 8 point algorithm). Both algorithms try to find a subset without incorrect points to get an improved estimate for the fundamental matrix.

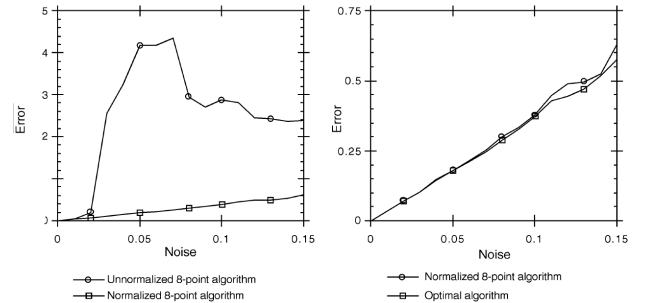


Figure 2: The 8 point algorithm compared with normalized 8 point and optimal algorithm (Image taken from [Har97])

The RANSAC and LMedS algorithms produce the best results. [Zha98] But as shown in [Har97] the 8 point algorithm performs almost as well or sometimes better as the best competing algorithm.

The prototype application uses the normalized 8 Point Algorithm to estimate the fundamental matrix because it produces good results and implementation is well documented in different publications. For a real application the RANSAC or LMedS algorithm should be considered.

2.3 Estimating the 3D position

With triangulation the difficulty is again dealing with measurement errors. The lines through the camera center O_n and projected points y_n intersect at the 3D position X of the light (Note the camera center is now in front of the camera, not behind it as in figure 1. This does not affect the algorithms however). If the measured points y'_n contain errors though, the lines will not intersect. See figure 3 for a graphical illustration.

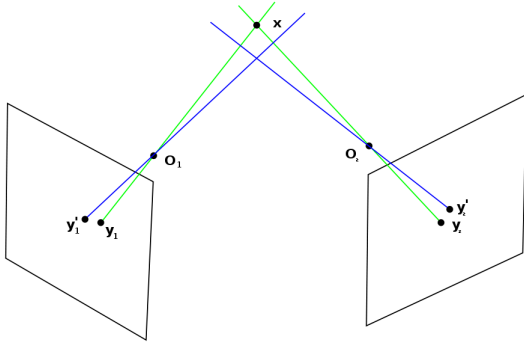


Figure 3: Measurement errors (image taken from wikipedia article on triangulation)

The triangulation algorithms assume the fundamental matrix is calculated with great accuracy compared to the errors in the measured points. This means if the lines through the camera center and measured points do not intersect, there is a measurement error not an inaccuracy in calculating the lines.

The measurement errors occur in where the camera sees points in the image plane. There are different reasons why these errors occur in the camera: errors in the digitization process, lens distortion, feature localization and more.

The scene is projected onto the cameras image plane in different ways. A good triangulation algorithm should be independent of the projection used. A naive algorithm like taking the midpoint of the place where the two lines come closest to each other, is not projection independent. [HS97]

In [HS97] there is a detailed explanation of triangulation were several approaches and algorithms are explained and compared. Below the most important algorithms are described.

- **Midpoint** The midpoint between the two lines through camera center and points is taken, were the two lines are closest to each other. This algorithm is not independent of camera projection and gives rather poor results.

- **Linear triangulation** These algorithms estimate the 3D position in different ways from a system of linear equations derived from $x = P.X$, which describes how point X is projected onto the image plane as x by camera matrix P . These algorithms are not independent of camera projection.

- **Polynomial** New points are determined which do make the lines though camera center meet. The points are defined as having the smallest squared distance to the measured points. The solution is found as the roots of a 6th degree polynomial. This method is optimal with regard to its assumptions.

For the prototype the algorithm was chosen for it's simplicity: linear triangulation (Linear-Eigen). In a real application, the polynomial algorithm should be used.

2.4 Interfacing the Wii Remotes

As mentioned before, the WiiMote controllers use the standard Bluetooth protocol (version 2.0) to interface with the Nintendo Wii or a workstation. Within Bluetooth they use a custom set of packets to transmits requests and data. Curious hackers have reverse engineered most of the content transmitted in these packets [Wii08]. Other projects have developed libraries for various programming languages to do the communication with the WiiMote [gl.08] [Cha08].

The WiiMote controllers has a whole scope of input and output features: 11 buttons, a 100hz IR camera operating with a 1024x768 resolution, a 3 axis accelerometer, LED lights indicating the player number, an audio speaker, vibration feedback and the option to attach an extension device (Nunchuck, classic controller).

The packets used to communicate with the WiiMote allow reading the input sensors, controlling the output devices, configuring reporting modes and more. Please see [Wii08] for detailed information about the WiiMote controllers features and how to use them.

In general using the WiiMote controllers worked well. There were a few minor problems, those are mentioned in the conclusions.

2.5 Software

The software prototype is implemented in Java. There are several pieces of functionality in the software, communication via Bluetooth with the wiimote, controlling the WiiMote using the custom protocol, gathering data and running the stereovision algorithms.

For Java there is a standardized API for Bluetooth applications [JSR06]. On Linux this API is implemented for example by the Avetana Bluetooth stack. We used this Avetana Bluetooth stack for the Bluetooth functionality.

To communicate with the WiiMotes another library was used that implements the specific functionality of a WiiMote: WiiRemoteJ [Cha08]. This library uses the Bluetooth library to communicate with the WiiMote.

The rest of the software was implemented our self. To explain a little about how it was implemented, the most important classes are discussed briefly.

- Class **WiiFinder** uses the WiiRemoteJ library to discover wiimotes. Once it finds a WiiMote it configures it. If enough WiiMotes are found, control returns to the main program.
- The **IRCollector** class does what the name suggests, it collects IR data as received from the WiiMotes.
- Class **Calibrator** uses gathered IR data to calculate the fundamental matrix.
- Class **Triangulator** uses the fundamental matrix and calculates the 3D position for IR points that are received.
- Class **Jogl** displays the results on screen. The image planes of both cameras are drawn next to each other. In them are the received position of the light and the epipolar line. Also it draws a 3D cube to visualize the calculated 3D position of the light.

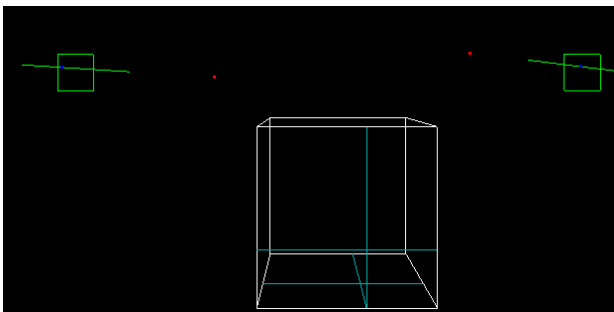


Figure 4: Screenshot of the prototype

See figure 4. In the top of the screenshot there are the green squares, which represent the image planes of the two cameras. The white perspective cube in the bottom half shows the estimated 3D position of the IR light.

In the image planes, blue dots show the measured position of the IR light. The red dots are the epipoles calculated during calibration. The green lines are the epipolar lines. Notice how the epipolar lines go through the measured position of the light towards the epipole.

See the estimated position of the IR light in figure 5 and compare to figure 4. The implementation of the triangulation algorithm in the prototype was buggy, sometimes the estimated position was wrong.

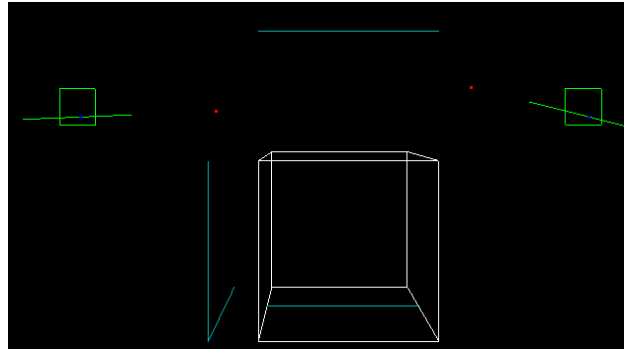


Figure 5: Triangulation error

3 Results

Though the implementation of the triangulation algorithm still had problems, the general approach used seems promising. With some work the presented stereovision techniques can determine the 3D position of a light. See 2.2 and 2.3 for a discussion of the algorithms.

Results and remarks from implementing the stereovision algorithms are below.

Calibration

The calibration process determines the relative geometry of the cameras and expresses that in a matrix called the fundamental matrix. This was implemented using the 8 point algorithm mentioned in section 2.2.

To check our implementation of the 8 point algorithm we prepared a sample data set and took an existing well known implementation from the OpenCV project [Ope08] to compare results.

The correctness of the fundamental matrix can further be checked by drawing the epipolar lines in the camera images. The recorded point should be on the epipolar line. See figure 4 for an illustration.

What was quite interesting, was that the fundamental matrix works best if it is calibrated using points in a large area of the visible space. For example, if the fundamental matrix is calculated with points close to each other, the epipolar lines drawn if the point is positioned at the edges of the visible space are inaccurate and don't go through the recorded point. The solution is obvious, move the point around the visible space while recording calibration points.

Triangulation

The triangulation calculates the 3D position of the point in the scene. The implementation of this algorithm was still buggy, the estimated 3D position was not correct.

WiiMotes

Using the wiimote controllers worked well, there were a few problems worth mentioning though.

- **Connecting** It was not always easy to connect multiple WiiMotes to the workstation. First one controller needed to connect and transmit some input data, and only then could the next controller reliably connect.
- **Disconnecting** Not terminating the Bluetooth connection explicitly could get the WiiMote into a blocked state. It would refuse connections until itself and the workstation were reset.
- **Data reporting** The collected data from the IR camera in the WiiMote is sent as a continuous stream of updates. However both WiiMotes can not transmit data simultaneously: sometimes a few packets from one controller are received and a few moments later a packet from the other controller. To be useful for the stereovision algorithms, points should be recorded with both cameras at the same moment in time. The solution was to discard points received more than 20ms apart.
- **Visibility** Sometimes only one camera can see the IR light or a camera sees multiple lights. The stereovision algorithms can only handle pairs of matching points, so other input needed to be discarded.

4 Conclusions

Using two WiiMote IR cameras and the discussed algorithms the 3D position of a light can be determined.

Calibration with the normalized 8 point algorithm worked well in the prototype. The LMedS and RANSAC algorithms mentioned in 2.2 should be considered for a serious application since they give even better results.

Even though the implementation of the linear triangulation algorithm in the prototype was buggy, results from literature show the approach works. For a serious application the polynomial algorithms discussed in 2.3 should be considered.

References

- [Cha08] Cha0s. Wiiremotej. <http://www.world-of-cha0s.hostrocket.com/WiiRemoteJ/>, last checked 2008.
- [FP02] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.
- [gl.08] gl.tter. Wiiyourself. <http://wiiyourself.gl.tter.org/>, last checked 2008.
- [Har97] Richard I. Hartley. In defense of the eight-point algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(6):580–593, 1997.
- [HS97] Richard I. Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding: CVIU*, 68(2):146–157, 1997.
- [HZ00] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
- [JSR06] Jsr-82. <http://jcp.org/aboutJava/communityprocess/final/jsr082/index.html>, 2006.
- [Lee08] Johnny Chung Lee. Wii projects. <http://www.cs.cmu.edu/~johnny/projects/wii/>, last checked 2008.
- [LF95] Q. Luong and O. Faugeras. The fundamental matrix: Theory, algorithms, and stability analysis, 1995.
- [Ope08] OpenCV. Open computer vision library. <http://www.intel.com/technology/computing/opencv/>, last checked 2008.
- [Wii08] WiiBrew. Wiimote. <http://wiibrew.org/wiki/Wiimote>, last checked 2008.
- [Zha98] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. *Int. J. Comput. Vision*, 27(2):161–195, 1998.