Excel@FIT 2015
http://excel.fit.vutbr.cz

Game Interface with Data Projector and Leap Motion

Jonas Holcner*

Abstract
The goal of this paper is to develop user interface using potential of the Leap Motion Controller in combination with projected image of a projector. This interface requires a projector calibration for user to be able to interact with the image. The calibration enables translation of points between coordinate spaces of Leap Motion and projector. Combined with data from Leap Motion, this allows user to interact with applications and games only by hands within the projected image. The solution contains a projector and a Leap Motion located over a table looking at the table desk. Leap Motion is used to track hand motions above the table. The capabilities of the proposed interface are demonstrated on a computer game, which was developed in Unity 3D engine.
In this paper is proposed a new way of interacting with applications and games using the unique control interface.

Keywords: Leap Motion — Camera calibration — Projector calibration — Leap Motion projector interface — Unity 3D

Supplementary Material: Demonstration Video — Downloadable Code

*xholcn01@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Lately, there has been an effort to come up with new ways of how people can control and interact with computers and applications. But not all of those prevail or live up to their full potential. One of those new controllers is the Leap Motion, a sensor, that tracks hand and finger motions and gestures. In this paper is described a user interface combining a projector and a Leap Motion.

The interface requires properly calibrated projector. The usual camera-projector set uses camera to capture calibration patterns. The problem here is that Leap Motion uses IR cameras which cannot see light emitted from the projector. Therefore we cannot use common calibration techniques. Instead, Leap Motion can be taken as pre-calibrated camera, that is able to capture hands instead of the calibration patterns.

There are many papers introducing different methods of calibrating camera - projector set. However, none of those delivers a way to do this calibration with a Leap Motion posing as a camera. One of the most renown papers on this topic is Zhang [1]. Zhang's calibration method requires a planar checkerboard grid with known dimensions to be placed in at least two orientations in front of the camera. It extracts corner points from each of these orientation and uses them to compute projective transformation between the n captured images.

This method is popular because of its ease of use and deployment. It does not require expensive scan-

Figure 1. Example of a Leap Motion usage

ning device but only an off-the-shelf camera and a projector. The checkerboard calibration pattern can be printed on any black-white printer.

G. Falcao, N. Hurtos and J. Massich [2] explains why we need to use already calibrated camera to calibrate projector. The projector cannot image the surface that it illuminates so the correspondence between the 2D projected points and the 3D illuminated points cannot be made without the use of a camera.

In the proposed solution, Leap Motion poses as a pre-calibrated camera. The calibration of projector is done using hands and fingers with Leap Motion tracking their location in space.

2. Projector calibration

Using projector calibration brings us a way to transform the points from 3D metric coordinates of the Leap Motion Controller to 2D points in pixels of the projector image. Calibration process consists of measuring points and computing intrinsic (projector's focal length and principal point) and extrinsic parameters (mutual position and rotation of projector and Leap Motion) of the projector from those points. From these parameters, we can compose matrices using which we can transpose the points from 2D projector pixel coordinate space into 3D Leap Motion metric coordinate space and the other way around. Based on this, we can modify the projected image.

As already stated, in our setup Leap Motion - projector, the Leap Motion poses as pre-calibrated camera, thus Leap does not need any sort of calibration. To calibrate the projector we need to establish set of pairs of corresponding 2D and 3D points. Set of 2D points in pixels of the projector image is defined on known positions. For each of them we need to find the corresponding 3D points from Leap Motion coordinate system. This can be done as shown in figure 2. The fingers tracked by Leap Motion then have to point on defined projected points in the image of the projector. This will give us the corresponding 2D points.

We will then compute the extrinsic and intrinsic parameters. The translation between 2D and 3D points is shown in equations 1 and 2.

$$s\,m' = A[R|t]M' \tag{1}$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

- A - intrinsic parameters matrix
- $[R|t]$ - extrinsic parameters matrix; relative position and rotation of projector and Leap Motion
- (X,Y,Z) - coordinates of a 3D point in the world coordinate space
- (u,v) - coordinates of the 2D projection point in pixels
- (c_x,c_y) - principal point that is usually at the image center
- f_x, f_y - focal lengths expressed in pixel units

An initial estimation of the intrinsic matrix can be used for the the calibration computation. These parameters are used in the initial intrinsic matrix:

1. c_x - half of the width of the projector's image in pixels
2. c_y - half of the height of the projector's image in pixels
3. f_x, f_y - focal length expressed in pixel units determined for the projector used in the interface (Asus S1)

The initial estimated intrinsic matrix (equation 3) determined for the projector Asus S1 (section 3):

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 962 & 0 & 400 \\ 0 & 962 & 300 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Next paragraph describes the process of calibration.

The prepared 3*3 matrix of 2D points is projected into the projector image as shown in figure 2. For each of those points, user has to put a finger on it for second after which the point disappears and the finger's location is saved. This gives us their corresponding position in Leap Motion 3D coordinates. Let's call process of projecting and scanning of 3*3 points a

calibration run. We can choose how many of those calibration runs should be carried out according to required precision. For the calibration to work not only in the projector image plane but above it, it is important to put finger on the projected points in different heights above the desk. The calibration process can be seen in Demonstration Video.

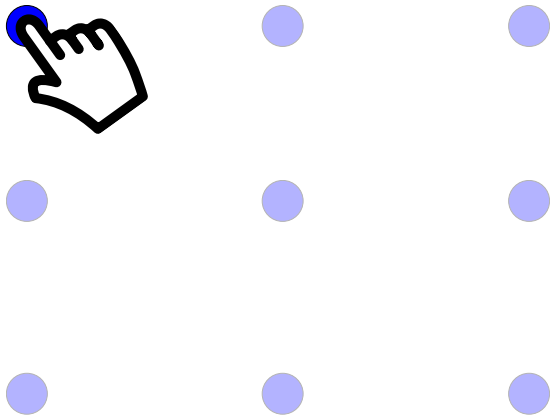

Figure 2. Example of the calibration process - user puts his finger on the projected points in order to get a corresponding points in space.

Easy way of finding out precision of the calibration is to translate position of user's index finger from Leap Motion coordinates into projector coordinates then draw a point on this position so that user can easily see how precise the final calibration is.

3. Hardware setup

Interface setup (figure 3) contains:

- Leap Motion Controller
- projector Asus S1, using resolution 800x600
- tripod for projector and Leap Motion
- desk covered with black canvas

Leap Motion is primarily intended to be used in lying vertical position, tracking hands above it. This way it gives the best results. Problem is if we want to use the Leap Motion against some kind of surface such as a desk. The issue is not the upside down position of controller but that it is used against a surface and not a empty area. Leap Motion uses infrared cameras and the light from them reflects from the surface of the table which results in too much distortion for Leap Motion to work properly. That is why important part of the setup is the black canvas which absorbs the excessive light and enables Leap Motion to work. Although black canvas is not very good surface to project onto as it absorbs the light emitted from projector as well.

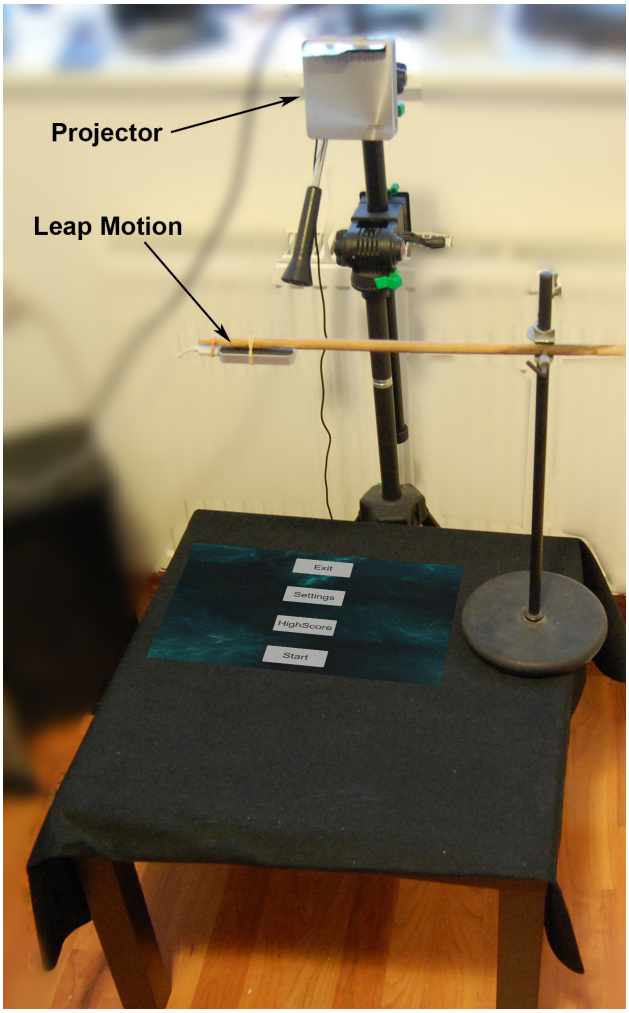

Figure 3. Interface setup - Leap Motion and projector both point on the table. After successful calibration, there will be workspace area between the table and Leap Motion in which user can control applications and games using his hands.

Both the Leap Motion and the projector must be stationary. Movement of any of those after successful calibration causes an incorrect result, because the calculated parameters of their mutual position would not be correct anymore.

4. Calibration software

The calibration software was developed in Python programming language. It uses PyQt library for the graphical interface. Main computation related with calibration is done using OpenCV library [3]. This language and libraries were chosen for their easy and flexible usage and ability to be run on all common platforms such as Windows and Linux.

To compute calibration matrices from input points, software uses OpenCV's method *calibrateCamera()*. Projecting points onto user's finger, used as a fast precision test, is done with *projectPoints()* method.

For more information about OpenCV calibration methods please read here [1].

5. Evaluation

To find out the exact precision of the calibration, we use reprojection error (more about reprojection error here [4]). Similarly to the calibration phase, software projects points on known coordinates in projector image. Instead of 3*3 matrix it uses 5*5 matrix. For each point, again the same way as during calibration, it gets corresponding point in 3D space. Then, for each of the 3D points it computes difference between where the point should have projected (the initially chosen position in projector image) and where the point projected using *projectPoints()*. Final reprojection error gives overall average error across all the measured points (equation 4).

$$err = \sqrt{\frac{\sum d(x_i, \hat{x}_i)^2}{n}} \tag{4}$$

- err - final reprojection error
- d - euclidean distance
- x_i - coordinate where the 3D point should have projected
- $\hat{x}_i$ - coordinates where the 3D point really projected
- n - count of all measured points

These are the results of reprojection error for different number of measured points (table 1). Problem with getting better precision is that it depends on user's ability to precisely put his finger onto projected points not only in plane of projector's image but also above it. Thus resulting coordinates from Leap Motion are only as precise as user can get.

Table 1. Average value of reprojection error in accordance with different number of measured pairs of points

Point count	9	27	45
Reprojection error (mm)	16.2084	10.5922	13.6603

Table 1 shows that the best precision i was able to achieve was when using 27 points as calibration input. Naturally precision should get better with more provided points. But after extensive testing best result was still achieved with 27 points when each calibration run was done in different height above the table. This is probably because, as already stated, this calibration is strongly dependant on user's precision which gets worse after many measured points. Nevertheless i plan to further address this problem in future testing.

[1] http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

Figure 4. Example of how the game is played. The spaceship moves in front of user's index finger.

6. Demonstrational game

I developed the game in order to show capabilities of the proposed interface. It was developed in Unity 3D engine which has means provided from its creators to use it in combination with Leap Motion. The game is based on this sample project [2] from which it uses all the needed models and sounds. It is a simple space arcade, in which the player controls a spaceship flying through different obstacles.

Every part of the game can be controlled using hand movements and gestures. The spaceship is positioned in front of user's index finger as can be seen in figure 4.

The game provides two different control inputs. The chosen type affects the way user interact with menu buttons and how he can shoot from the space ship. I plan to find out which one is more user friendly after user testing.

- *Clicking* - This type of control input is common for modern devices such as smartphones, tablets and other touch screens. Therefore it could be intuitive for users.
- *Pointing* - Due correct calibration, the whole empty space above the projector image can be

[2] http://unity3d.com/learn/tutorials/projects/space-shooter

used for interaction. This control input provides the user a way to activate a menu button without touching it. User only needs to point onto it for a while (0.7 sec in default setting) to active it, in any height above the table.

Design of the ingame menu was done in accordance with [3]. Menu buttons are large and user gets visual feedback when selecting it. Because this interface is used differently, menu buttons are arranged in reverse order to the usual one as shown in figure 5. The buttons closer to the user should be easier to reach, so more frequently used buttons are at the bottom of the screen and not at the top.

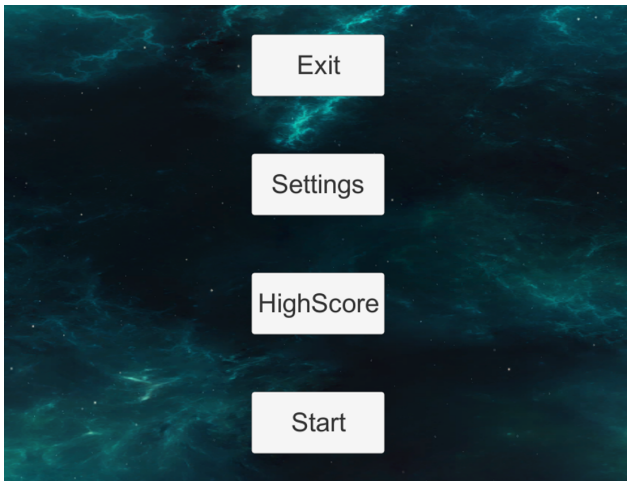

Figure 5. Ingame menu - the buttons are in reverse order than usual. That makes the more often choices more accessible for users.

The game can be seen in Demonstration Video.

7. Conclusions

In this paper was introduced a new kind of control interface using the Leap Motion Controller and a projector. The interface allows to control games and applications using user's hand directly within the projected image. This is achieved with unique projector calibration on which is the interface based.

The smallest achieved calibration error is about 10.5 mm. The precision of the calibration strongly depends on the precision of the measured input points.

Black canvas is used to deal with an excessive light from the Leap Motion that leads to problems with using Leap against a desk surface. The developed software including both the calibration and the game is freely released to be experimented with.

[3] https://developer.leapmotion.com/documentation/csharp/practices/Leap_Menu_Design_Guidelines.html

Interesting way to continue the research would be to increase the calibration precision by using better input of 3D space coordinates than is user's hand for instance by using robotic arm. Next, it would be useful to solve the detection problem against surfaces in some other way than by using black canvas, because it absorbs not only Leap's light but also the light emitted from the projector.

Acknowledgements

I would like to thank my supervisor Jiri Zahradka for guidance and useful comments.

References

[1] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, November 2000. http://research.microsoft.com/en-us/um/people/zhang/Papers/TR98-71.pdf/.

[2] Joan Massich Gabriel Falcao, Natalia Hurtos. Plane-based calibration of a projector-camera system. 2008. https://procamcalib.googlecode.com/files/ProCam_Calib_v2.pdf.

[3] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000. http://opencv.org/.

[4] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.