Laser pointer drawing

Tomáš Pokorný (xtompok@gmail.com)

Abstract

Project Laser pointer drawing is about computer processing images captured through the webcam. In image from camera is found lighted point, coordinates of this point are transformed to dataprojector coordinates and they are used to draw on screen or do something else. Application is written in Objective C and uses Mac OS X libraries.

Equipment

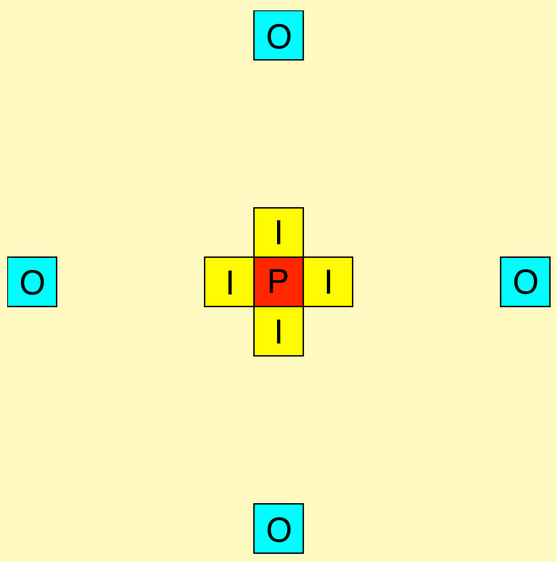
This equipment is needed to run the project:

- Mac
- webcam
- dataprojector
- laser pointer

The project uses CocoaSequenceGrabber for getting images from camera, so it works best with the integrated iSight. Working with externals camera is not tested. Used resolution of the camera is 320×240. Recommended Mac has 2 GHz processor to process images fast enough. Used resolution of dataprojector is 800×600 and the color of the laser pointer is red.

Processing image

Images are captured by framework CocoaSequenceGrabber. We need to find the lighted point, if it is on image. To find it, it is used cross method. In this method there are three types of points:



- Central point P, which is the current tested point
- Inner points I, whose are on the top, bottom left and right of the central point.
- Outer points O, whose are five pixels on the top, bottom, left and right of the central point and represents the background

The central point goes point by point through all image and computes average of inner and outer points, whose are aviable and compare them with configured values. If it's lower than in configuration, they are discarted. From not discarted points is chosen point with maximal values.

References

[1] D.-K. Kim, B.-T. Jang, C.-J. Hwang A Planar Perspective Image Matching using Point Correspondences and Rectangle-to-Quadrilateral Mapping in *Image Analysis and Interpretation, IEEE Southwest Symposium 2002*

[2] B. Andersen How to Resize an NSImage [online]. 2010 [cit 2010-11-10]. Aviable on WWW: http://weblog.scifihifi.com/2005/06/25/how-to-resize-an-nsimage/

Transformation

We need to transform coordinates from image to projector, because the axis of the camera isn't parallel with axis of the screen, so the image is distorted – the screen is not rectangle on the image from camera, but some quadrilateral. To transform it to rectangle is used perspective transformation, where each point is transformed with these equations:

$$x' = \frac{a_0 x + a_1 y + a_2}{a_6 x + a_7 y + 1} \quad (1)$$

$$y' = \frac{a_3 x + a_4 y + a_5}{a_6 x + a_7 y + 1} \quad (2)$$

The transformation could be written to matrix:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \cdot \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ a_6 & a_7 & 1 \end{bmatrix}$$

To get transformed values of x and y we need to normalise them:

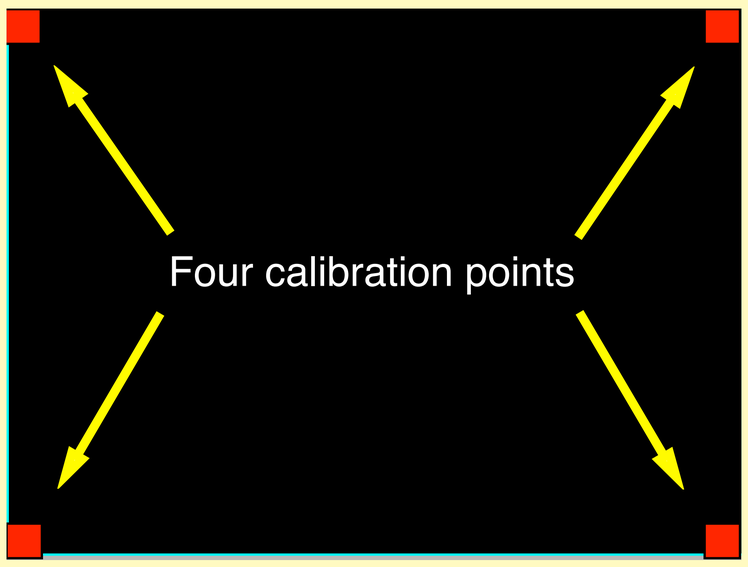
$$x' = \frac{u}{w}$$

$$y' = \frac{v}{w}$$

There were some other transformations, such as 2 point transformation or ratio transformation, which wasn't so accurate, so now they aren't used, but they were invented only by members of this project.

Calibration

To get the right values of $a_0, \ldots, a_7$ we need some calibration points. For this reason, there is an calibration. After start of calibration, four calibration points are showed in sequence by dataprojector in corners of drawable area.



For each point program finds its position in image using cross method from Processing image. After all points are showed, coordinates of their positions are sent to transformation classes, which computes its constants for transformation.

Viewing

There are many possible ways how to visualise results, some of them are mentioned below.

Single color drawing

On the current position of laser pointer is showed cross and the position is added to the `NSBezierPath` object which is drawed with green color. If there isn't a point from laser pointer, cross isn't drawed and the path is diconnected. If the light is back again, virtual "pen" is moved to new position and starts drawing again. The width of "pen" could be changed from GUI. From GUI could be also reseted a drawing.

Multi color drawing

Principle of this drawing is the same as in single color drawing. There are some aditional things – three squares with three colors. If the point from pointer is in this square, color of the virtual "pen" changes to the color of square. There are two other squares on the screen – resetting square and pausing square. Resetting square is placed in the lower left corner of the screen and if the pointer is pointing onto it for twenty processed images, the image is reseted. The pause square is square placed under the color setting squares and if it's pointed onto it, drawing image is paused until is pointed on it again.

Exporting to SVG

The saved drawed image

should be converted to the simple SVG file using Perl script `bezpath2svg.pl`

Thanks

Thanks to Marek Blaha, Tomáš Gavenčak, Lukáš Langer, Jan Olšina and other people, who helped me with the project.