Virtal 3D & Hough transform
Computer Vision, Lab Exercise 3
Andreas van Cranenburgh (0440949), Willem Bressers (5687063)
January 28, 2008

Abstract

This report discusses the implementation of a virtual 3D geometry projected onto a $2D$ image, as well as the detection of lines through a Hough transformation.

1 Problem specification

Calibrating a camera requires knowledge of the way 3D coordinates map on the resulting 2D output of the camera. Using this information a projection matrix can be constructed to convert 3D into 2D coordinates.

The second problem in this exercise concerns the detection of straight lines in an image. The edges of an edge detector only constitute a set of pixels. The Hough transform is able to approximate lines described by these pixels.

2 Theory and implementation

2.1 Camera

2.1.1 Pinhole camera

In a $3D$ world the coordinates are represented as $(X, Y, Z)^T$ whereas the coordenates in a $2D$ world are represented as (x, y). Because of the projection, the coordinates in both worlds needs to be homogenous coordinates. The relation between the 2 coordinate systems is:

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix} = M \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{1}$$

Where M is the $(3x4)$ camera matrix.

2.1.2 Estimating a camera's projection matrix

The 3D real world coordinates (X, Y, Z) represent all coordinates in the real world. But an image is 2D so the real world coordinates need to be translated to an image with 2 dimensions. Such a translation can be done by the matrix M which is called the "camera" or "projection matrix". The matrix can be multiplied by real world coordinates which results in a vector cointaining $(\lambda x, \lambda y, \lambda)^T$. The matrix M can be divided in 2 different matrices. M_{int} represents the internal variables of the matrix and M_{ext} represents the external variables of the camera.

$$M_{int} = \begin{pmatrix} f_x & s & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2}$$

The internal matrix M_{int} is an orthogonal matrix based on the focal parameters f_x and f_y, the shear s of the CCD-chip, and the offset parameters u_0 and v_0. Most of the camera matrices assume that f_x and f_y are 1, and that s, u_0 and v_0 are 0.

$$M_{ext} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

The external matrix M_{ext} is also an orthogonal matrix based on the camera's distance to the object and the rotation of the camera. The values r_{11} to r_{33} are calculated using the world coordinates (X, Y, Z). The values t_x, t_y, t_z represent the world origin in camera coordinates.

2.1.3 Projecting cubes into a picture

Before the projection matrix can be used, its variables need to be bound to values. This is the hard part because it is tough to read 3D coordinates with any precision from an image. This problem can be solved by using a photo of two chessboards.

The chessboards have a grid of known size, which forms the basis of a simple coordinate system. Two chessboards put next to each other at an angle of 90 degrees makes for a perfect calibration system. Now our variables can be arrived at by some point-and-click action performed by the user. For example, the user can be asked to start at $(0, 0, 0)$, go to $(3, 0, 0)$ etc. each time taking three steps back and forth for each axis.

Using such information a projection matrix can be calculated by using the SVD, similar to doing projective transformations, except with one extra coordinate for 3D. Figure 1 shows 2 chessboards with 3 cubes projected on them.

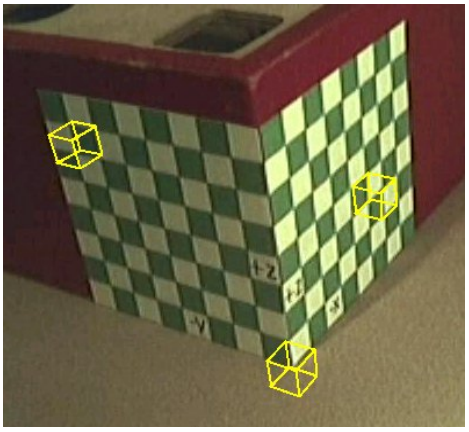

Figure 1: Cubes projected on chessboards. These were projected using given calibration points.

3 Hough transformation

3.1 Hough space

The Hough transformation detects features that can be described by a function, eg. lines. The first step of the Hough transformation is to detect edges, for example using the Canny edge detector. Then each edge coordinate is parametrized into a sine curve, representing all lines intersecting the point. This sine curve is mapped into a matrix with each pixel representing the distance to the original edge, combined with a Gaussian blur, to accomodate for imprecesion. The axes of the sine curves are ρ and ϕ, which represent the distance and angle to the origin, respectively. All of the sine curves are summed after which clusters form at intersecting since curves. This is the Hough space.

3.2 Clustering

Detecting these clusters can be done using statistical algorithms, such as k-means clustering. It is also possible to ask the user to point and click at a plotted histogram, to help partition the Hough space. This method proved unsatisfactory. The clusters are now detected by applying a threshhold and shrink operation, consecutively. The shrink operation reduces blobs to single pixels, in the center of the blob. Their coordinates are used to find the local maxima in their 11×11 neighborhood.

3.3 Dehoughing

The coordinates of the maxima just obtained represent ρ and ϕ pairs. In order to convert them back to image coordinates we used these formula:

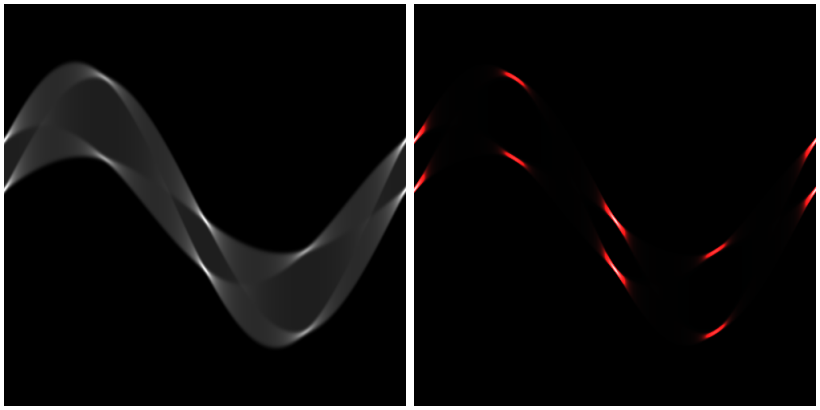

Figure 2: The left image shows the Hough space. The image on the right has been threshholded and adapted to better emphasize its maxima.

$$\begin{aligned} y &= ax + b \\ a &= \tan -\phi \\ b &= \frac{r}{\cos \phi} \end{aligned} \tag{4}$$

Now in order to calculate the point P where two lines intersect, the cross-product is applied to two line vectors, after augmenting them with a homogeneous coordinate:

$$\lambda P = \begin{pmatrix} a_1 \\ 1 \\ b_1 \end{pmatrix} \times \begin{pmatrix} a_2 \\ 1 \\ b_2 \end{pmatrix} \tag{5}$$

Determining the right threshholds so as to obtain a reasonable number of lines is a laborous task, as we have not found a stable way to determine it. Also, the order in which they occur in the Hough space is not always the order that the quadriliteral should have, which results in an hourglass-shaped projection, or a mirrored projection (see figure 3).

A single point in Hough space represents a straight line in the image. Each time a sine curve is added to the Hough space, its pixels "vote" for the lines on that sine curve.

4 Optimizations and implementations

A possible optimization is taking the gradient direction into account, as produced by Canny edge detection. This can reduce theamount of stray lines in the Hough space significantly, according to Wikipedia.

5 Usage

The function `testCamera` demonstrates the virtual camera viewing a cube. `testCameraProjection` demonstrates the projection of cubes onto an image using given calibration points. `testProjectionCalibration` is similar, but allows the user to click on the points to be calibrated.

The function `testHough` runs the Hough transformation on a "`perla.tif`" and detects its corners, after which it applies a projective transformation using these corners.

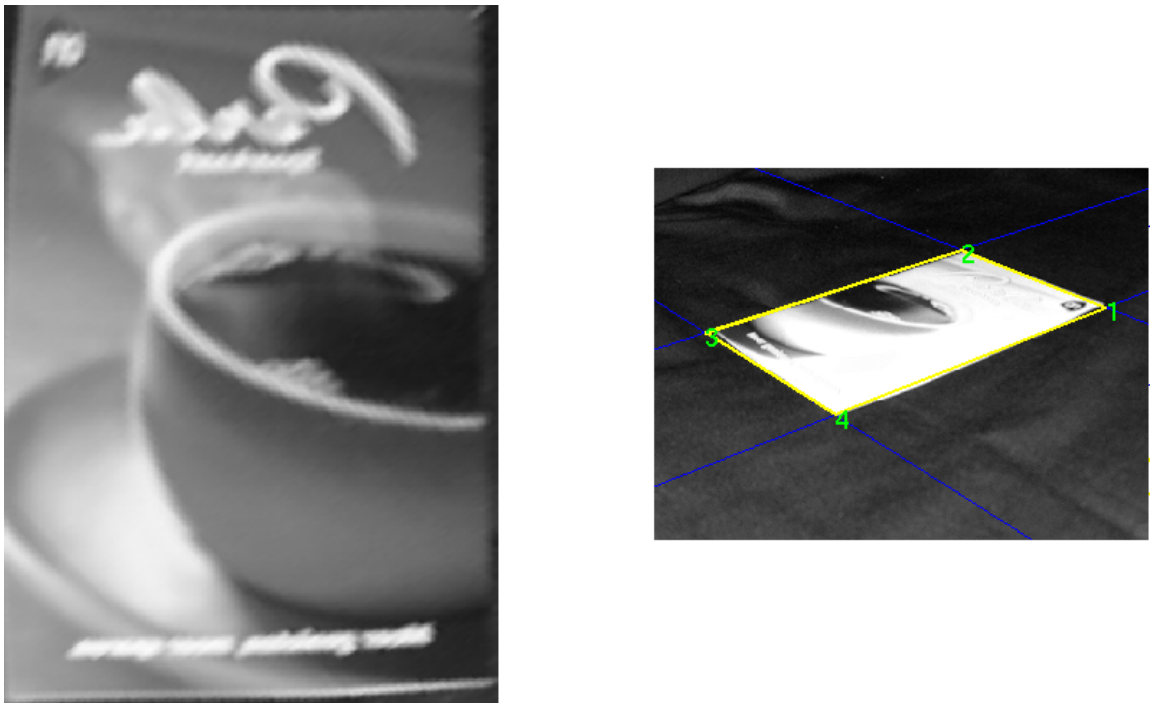

Figure 3: Projective transformation with detected lines. Note that the image is mirrored reflecting the order of the detected lines.