1 Lens Distortion

- *Algorithm:* Lens Distortion Correction (algo. 1)
- *Input:* An image with distortion
- *Complexity:* $\mathcal{O}(n)$, n is the number of pixel
- *Data structure compatibility:* Image
- *Common applications:* Professional photography, image restoration in MR, computer vision, digital image processing.

Lens Distortion

Given an image with distortion, correct the distortion and output an image with the distortion fixed.

Description

Due to the defect in wide-angle lens or telescope lens, the image of the line from a photograph will not be straight when it is on the photograph. And this is called a distortion. The correction of such a distortion then becomes an important topic in image calibration. In this problem we will discuss about the radial distortion, which includes two types: barrel and pincushion distortion.

The radial distortion can be measured with the Brown model,

$$\begin{pmatrix} x_{res} - x_c \\ y_{res} - y_c \end{pmatrix} = L(r) \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} \tag{1.1}$$

(x_{res}, y_{res}) is the correct position of each pixel (x, y), and $L(r)$ is a function on $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$, (x_c, y_c) is the center of the image.

To help restore the image, there are four stages.

1. Canny method: detect an edge.
   In this stage, the canny method will detect the edge. The whole process can be found in the reference part. [1]

2. Hough transformation: distortion parameter estimation.
   For Hough transformation, we will restrict the function $L(H)$ in equation 1.1 as

$$L(r) = \frac{1}{1 + k_1 r^2 + k_2 r^4 + \cdots + k_i r^{2i} + \cdots}$$

   And we will get

$$r_{res} = \frac{r}{1 + k_1 r^2} \tag{1.2}$$

Note here r_{res} is the distance from the center after applying the model. Since the original Hough transform will separate a line segment into several, which will not help the image restoration, an improved hough transformation is introduced. The algorithm of this improved hough transformation is described in algorithm 1.

3. Distortion parameter optimize.

   In this stage, the previous parameter obtained from the hough transformation will be optimized by removing a certain number of trivial edges, which is less-visited. In the second algorithm, the main purpose it to minimize the error from previous algorithm 1.

4. Image correction. Now we have the correction parameter, we will use the equation 1.2 to obtain

$$(1 + k_1 \cdot r^2) r_{res} - r = 0$$

   Solving this equation and we can get the corrected image.

Algorithm 1: Line detection with hough transformation

Input : Image coordinate, Image and maximum expected line segments N
Output: parameter p_0 for normalized distortion, most used lines

Function `line_detection`(*Image, N*)**:**
  for $p = \{p_{min} + n\delta\}$, $p \leq p_{max}$ **do**
    for (x, y) *in image* **do**
      /* α is the orientation parameter */
      Update α according to the old coordinate and old orientation parameter;
      Mark the new α as α_{new};
      for $\beta \in [\alpha_{new} - d_\alpha, \alpha_{new} + d_\alpha]$ **do**
        /* Update new distance */
        $d_l = |\cos\beta x_{res} + \sin\beta y_{res} + d|$;
        weight of selected line with parameter $\{dist, \beta, p\} + \frac{1}{1+d_l}$;
      end for
      /* Sum to be all weight */
    end for
    $sum = \sum weight(d, \alpha, p)$;
    if $sum > -1$ **then**
      $p_0 = p$;
      $max_{Hough} = sum$;
    end if
  end for
  return p_0;
end

Algorithm 2: Optimize p_0

Input : p_0
Output: p_{opt}

Function `Optimize`(p_0)**:**
- $p_{opt} \leftarrow p_0$;
- $\sigma = 1$;
- $p = p_0 + tolerence + 1$;
- **while** $|p_{opt} - p_0| > tolerence$ **do**
  - /* $E()$ is the expectation function */
  - **while** $E(p) > E(p_{opt})$ **do**
    - $\sigma = \sigma \cdot 10$;
    - $p_{new} = p_{opt} - \frac{E(p_{opt})}{E(p_{opt})+\sigma}$;
  - **end while**
  - $\sigma = \sigma/10$;
  - $p = p_{opt}$;
  - $p_{opt} = p_{new}$;
- **end while**
- **return** p_{opt}

end

Algorithm 3: image restoration

Input : Image and lens model
Output: Perfect image without distortion

Function `image_restoration`(*Image, parameter r*)**:**
- **for** $(x_{res}, y_{res}) \in$ *Output image* **do**
  - $r_{res} = \sqrt{(x_{res} - x_c)^2 + (y_{res} - y_c)^2}$;
  - $id = \lfloor r_{res} \rfloor$;
  - $wei = r_{res} - id$;
  - /* Vector contains result from $r = \frac{1-\sqrt{1-4k_1(r_{res})^2}}{2k_1 r_{res}}$ */
  - $r = (1 - wei) \cdot Vector[id] + wei \cdot Vector[id + 1]$;
  - $(x, y) = (x_c + r\frac{x_{res}-x_c}{r_res}, y_c + r\frac{y_{res}-y_c}{y_res})$;
  - **return** *The image built with* (x_{res}, y_{res});
- **end for**

end

Something about time complexity: The time complexity of this algorithm shall be analyzed step by step.

1. Edge detector: canny method. In this step, the time complexity is $\mathcal{O}(N)$, where N is the number of pixels.
2. Hough transform: In this step, the time complexity is $\mathcal{O}(N_{lines})$
3. Distortion parameter optimization: In this step, the time complexity is given by $\mathcal{O}(N_{edges})$
4. Image correction: In this step, the time complexity is given by $\mathcal{O}(N_{pixels})$

Since $N_{pixels} >> N_{lines} >> N_{edges}$, the final time complexity is given by $\mathcal{O}(N_{pixels})$

References

- J. Canny, A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8 (1986), pp. 679698. http://dx.doi.org/10.1109/TPAMI.1986.4767851.
- M. Aleman-Flores, L. Alvarez, L. Gomez, and D. Santana-Cedres, Automatic Lens Distortion Correction Using One-Parameter Division Models, Image Processing On Line, 4 (2014), pp. 327343. http://dx.doi.org/10.5201/ipol.2014.106.
- L. Alvarez, L. Gomez, and J.R. Sendra, Algebraic Lens Distortion Model Estimation, Image Processing On Line, 1 (2010). http://dx.doi.org/10.5201/ipol.2010.ags-alde.