Machine vision

Summary # 1

Humans perceive the 3D world around them with apparent ease:

- You can count the number of people in a room with no problems
- You can guess people’s emotions from their facial expressions

Researchers have been trying for decades to understand human vision system. In parallel, researchers in computer and machine vision have been trying to mimic the human vision system, and develop mathematical models. Recently there have been many advances in machine vision. So far a computer cannot interpret scenes better than a two-year old.

I. A LITTLE BIT OF HISTORY

In 1966 a professor at MIT asked a student to connect a camera to a computer and let the computer describe what the camera sees (this was a summer project). It turned out that this problem was more difficult than what the professor thought. Machine vision research began in 1970s by researchers in robotics and artificial intelligence. The field of image processing is an older well developed field. The goal in image processing is to transform the image to another image (more useful form), for example compression, image enhancement, noise removal are part of image processing. While in machine vision, the goal is to recover information and interpret it, this task is left to the user in image processing. Thus, image processing techniques can be used in the early stages of machine vision. Research continues in machine vision and targets more difficult and complex problems such as learning and situational awareness.

Why is machine vision a difficult problem?

- It is very complex
- It is an inverse problem because it aims at producing 3D models from image data, inverse problems are more difficult in general.
- Computer graphics where the goal is to produce image data from 3D models can be seen as the equivalent forward problem.

II. COMPUTER AND MACHINE VISION APPLICATIONS

We can cite among many other applications:

- optical character recognition
- handwritten postal code
- automatic license plate recognition
- medical imaging
- automated surveillance
- fingerprint recognition
- face recognition
- autonomous robots
- quality control

III. IMAGE SENSOR: CCD AND CMOS

The most important element in the camera is the “image sensor”. There exist two different technologies for digital cameras:

- CCD: charge coupled device
- CMOS: complementary metal oxide semiconductor
- CCD technology began in 1970 at Bell labs. Thus it is more mature.
- CMOS technology began also in 1970s but it became practical only in the 1990s.
- The basic working principle for both technologies is the same: convert light to electrons and then to a digital signal. The main difference is in the readout process.
- In CCD the circuitry is in the (separate) PCB. In CMOS the circuitry is in the image sensor itself.
- CMOS is more noisy because of the additional circuitry
- The readout speed is faster for CMOS
- CMOS and CCD are sensitive to lights ranging from near ultraviolet to near infrared.
- Is it possible to do infrared imaging using CMOS and CCD technologies? Yes. Look at the spectrum compared to the human eye.

The sensor size is an important element: several conventions are used to describe the size of the image sensor in a digital camera.

What are the most important characteristics of a camera?
Answer: noise, cost, speed, etc....

IV. IMAGE REPRESENTATION

An image is represented by a continuous function \( f(x, y) \) where

- \( x, y \) are the spatial coordinates
- \( f \) is proportional to the brightness (gray level).

Note that function \( f \) and the spatial variable have limited values. A digital image (image stored in a computer) has the form of a matrix: \( I(i,j) \).

A. Sampling and quantization

Because an analog image is a continuous function of two continuous variables, two operations are needed to obtain a digital image from the analog function: sampling and quantization.

1) Sampling: Convert the continuous space to a discrete space

\[ (x, y) \leftrightarrow (i, j) \]
2) Quantization: Convert the continuous gray level function to a discrete gray level

- Sampling affects the number of pixels. The notation:

\[ M \times N \text{ pixels} \quad (2) \]

is used for the size of the image (in terms of the number of pixels). Usually \( M \) and \( N \) are power of 2.
- Quantization affects the number of gray levels.
- 256 levels of gray is widely used, where
  - zero corresponds to black
  - 255 corresponds to white
- For 256 levels of gray how many bits/pixel we have? The answer is 8 bits/pixel.
- Pixel stands for “picture element”
- Voxel (volume element) is used in 3D imaging
- The size of the image data is characterized by the number of gray levels and the size of the image (number of pixels)

Digitization allows to convert the continuous 2D function representing the analog image to a digital image:

\[ f(x, y) \rightarrow I(i, j) \quad (3) \]

The two operations used to get the digital image from the original image: sampling and quantization as shown in the table below.

<table>
<thead>
<tr>
<th>digitization</th>
<th>continuous</th>
<th>discrete</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampling</td>
<td>((x, y))</td>
<td>((i, j))</td>
<td>pixel coordinates</td>
</tr>
<tr>
<td>quantization</td>
<td>(f)</td>
<td>(I)</td>
<td>intensity</td>
</tr>
</tbody>
</table>
After digitization (sampling and quantization), we have a matrix of numbers:

$$
\begin{bmatrix}
I(1,1) & I(1,2) & \ldots & I(1,N) \\
\vdots & \ddots & \vdots \\
I(M,1) & I(M,2) & \ldots & I(M,N)
\end{bmatrix}
$$

(4)

The image has $M$ rows and $N$ columns. In Matlab the origin is $I(1,1)$ (top left).

- For 256 levels of gray, the transition is smooth; the human eye cannot detect the difference between two successive levels of gray. If for example we use 4 bits, the human eye will perceive the jumps or discontinuities when intensity changes from one level to the next one. An illustration is shown in figures 3 and 4.

V. BASIC OPERATIONS

We can perform operations on the pixel value (intensity level), or pixel coordinates. We assume $K$ levels of gray, i.e., gray level goes from 0 to $K - 1$.

- Notation: here, we use $f(i,j)$ for the digital image instead of $I(i,j)$. Possible operations are listed below.
- Image offset:
  $$g(i,j) = f(i,j) + b$$
  (5)
  where $b$ is called the bias; it can be used to control the brightness of the image.
  - $b > 0$: brightened version of the original image
  - $b < 0$: dimmed version of the original image
  - Saturation may happen for large values of $b$.
- Image scaling
  $$g(i,j) = af(i,j)$$
  (6)
  where $a$ is called the gain. It can be used to control the contrast.
- Figure 5 shows controlling the brightness and the contrasts using offset and scaling operations.
- Image negative
  $$g(i,j) = -f(i,j) + K - 1$$
  (7)
- Nonlinear point operation: It is also possible to apply a nonlinear function to the image:
  $$g(i,j) = h(f(i,j))$$
  (8)
  where $h$ is the nonlinear function. Example:
  $$g(i,j) = \Log(1 + f(i,j))$$
  (9)
  or
  $$g(i,j) = [f(i,j)]^\gamma$$
  (10)
  where $\gamma$ is a constant. The effect on nonlinear functions on images can vary widely. This is illustrated in figure 6.
- Operations between images:
  - Addition
    $$g(i,j) = f_1(i,j) + f_2(i,j)$$
    (11)
  - Subtraction
    $$g(i,j) = f_1(i,j) - f_2(i,j)$$
    (12)
  In this case, $f_1$ and $f_2$ must have the same size. Subtraction can be used to detect the difference or changes in an image. Addition and subtraction are shown in figure 7.
- Linear blend operator
  $$g(i,j) = (1 - a)f_1(i,j) + af_2(i,j)$$
  (13)
  where $a$ varies between 0 and 1, it is called the blending
Machine vision

Original image, $f$

$g = f + 100$

$g = f - 100$

$g = 3f$

Fig. 5. Illustration of image offset and image scaling

Original image

Log transformation

$[f(i,j)]^{1/2}$

Cosine transformation

Fig. 6. Illustration of some nonlinear operations on an image

ratio. It determines the influence of each input image on the output. The output image is a linear combination of the input images. The linear blend operator allows to perform a spatial cross-dissolve between two images.

A. Geometric operations

Geometric operators are used to modify the pixel location (coordinates) not its value

- Image translation

$$g(i,j) = f(i-a,j-b) \quad (14)$$

- Image rotation:

$$i_2 = i_1 \cos(\theta) - j_1 \sin(\theta)$$

$$j_2 = i_1 \sin(\theta) + j_1 \cos(\theta) \quad (15)$$

allows to rotate the image by angle $\theta$

- Example 1: $\theta = 0$ : $i_2 = i_1$, $j_2 = j_1$

- Example 2: $\theta = 180$ : $i_2 = -i_1$, $j_2 = -j_1$

B. Image histogram

- Image histogram $H_f$ is the graph of the frequency of occurrence of each gray level in $f$. The histogram plots the number of pixels for each intensity value. It allows to view the intensity distribution in an image.

- $H_f$ is a 1D function. If we write:

$$H_f(k) = P \quad (17)$$

then $P$ is the number of times gray level $k$ appears in image $f$.

- For a given image, is the histogram unique? Yes

- Is it possible for more than one image to have the same histogram? Yes

- I have a histogram, can I obtain the original image? No

C. Histogram equalization

The goal is to improve the quality of the image by obtaining a flat or near flat histogram. There exist different simple approaches such as

- Look at the darkest and brightest pixels and map them to pure black and pure white, respectively.

- Find the average and push it towards the middle.

D. Data classes for image representation

Several data classes are supported by Matlab:
• uint8: unsigned 8-bit integer in the range [0, 255]
• uint16: unsigned 16-bit integer in the range [0, 65535]
• double: scaled to [0, 1]
• logical: values are 0 or 1.

E. Image types:

Matlab supports different types of images:
• Intensity images
• Binary images
• Indexed images
• RGB images

More details at:

F. Converting between classes

There are some functions that allow to convert from one class to another one.
• Example: `im2uint8` can be used to convert logical, uint16, double to uint8
• Example: `im2bw` can be used to convert uint8, uint16, double to logical (binary)

Entire list can be found at:

G. Some Matlab functions

• read an image: `f=imread('filename')`
• display an image: `imsho(f), image(f), imagesc(f)`
• write an image: `imwrite(f,'filename')`
• size of an image: `size(f)`
• information about image file: `imfinfo filename`

H. Image entropy

Entropy is a statistical measure of randomness in an image, it is a measure of disorder. It can be used to measure the quantity of information in an image. Entropy is given by

\[
E = - \sum P(f) \log(P(f))
\]  

where \( P \) is a probability in general. For images, \( P \) is obtained from the histogram.

• Example 1: Assume a 4–level, 3 by 3 pixels image as shown below:

\[
\begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 1 \\
2 & 3 & 2 \\
\end{bmatrix}
\]
The probability table is given by:

\[
\begin{array}{c|c}
 f & P(f) \\
 0 & 3/9 \\
 1 & 3/9 \\
 2 & 2/9 \\
 3 & 1/9 \\
\end{array}
\] (20)

from which
\[
E = -\left[\frac{3}{9} \log\left(\frac{3}{9}\right) + \frac{3}{9} \log\left(\frac{3}{9}\right) + \frac{2}{9} \log\left(\frac{2}{9}\right) + \frac{1}{9} \log\left(\frac{1}{9}\right)\right] = 1.31
\] (21)

• Example 2: Assume our image is now given by

\[
\begin{array}{ccc}
 & & 1 \\
0 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
\end{array}
\] (22)

The probability table is given by:

\[
\begin{array}{c|c}
 f & P(f) \\
 0 & 5/9 \\
 1 & 4/9 \\
 2 & 0/9 \\
 3 & 0/9 \\
\end{array}
\] (23)

from which
\[
E = -\left[\frac{4}{9} \log\left(\frac{4}{9}\right) + \frac{5}{9} \log\left(\frac{5}{9}\right)\right] = 0.687
\] (24)

Clearly the first image has more information.