Machine Vision Processing System based on Embedded Linux

C. MA
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

L.P. XU
Yantai Vocational College, Laishan Yantai City, Shandong, China

Z.D. WANG, K. HE
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
Shenzhen Key Laboratory of Precision Engineering, Shenzhen, China

R.X. DU
The Chinese University of Hong Kong, Hong Kong

ABSTRACT: Machine vision is widely used in automatic equipments, but the vision system based on embedded Linux is seldom at present. The key points and difficulties are the image processing and its real-time on embedded system. This paper introduces the vision processing system based on ARM and Linux, mainly introduces the USB camera driver, image matching, camera calibration and object localization. Comparing the conventional method using PC with universal software, vision processing system based on ARM and Linux greatly reduces the system size and cost. Experimental results show that this system can accurately recognize and locate the target object. The efficiency can meet the requirement basically, and it still has room to improve. Combined with industrial robot, this vision system can be used to automatic carrying, assembly, etc.

KEYWORD: machine vision; template match; camera calibration

1 INTRODUCTION

With the development of industrial automation, machine vision system is widely applied in automatic equipments. Currently, the conventional method is PC or IPC (industrial computer) with universal image processing software. The IPC brings the large size and high cost. And the universal software has multiple functions and good performance, such as the HexSight of Adept Company and the CVB of Stemmer Imaging Company. But the software structure is complicated and difficult to master. Usually only a limited amount of functionality is used and results in great waste.

In contrast, ARM controller has small size and its efficiency is improved gradually with the development of IC chip. The Linux system used on ARM is open source and has steady and flexible OS kernel. The machine vision on ARM and Linux system will have broad applications and it is a general trend. Although the vision system based embedded Linux had been seen in some literatures, the application in industry is seldom. Paper [1-2] only introduce the image capturing and display of USB camera, the image processing and object localization are not mentioned.

The key procedure in machine vision system includes the industrial camera driver, camera image processing, camera calibration, object recognition and localization. In the rest of the paper, we will introduce the system in detail.

2 KEY TECHNOLOGIES OF MACHINE VISION SYSTEM

2.1 *USB camera driver based on libusb*

The USB camera is used in this system for its advantages of rapid speed, plug and play, flexible interface, etc. The first task is the implementation of USB device driver. The commonly used technology is based on kernel driver technology[3]. But it has some questions. For instance, the developer should be familiar with the Linux kernel system, and the driver is incompatible, it should change some program when the kernel version updated.

Libusb (www.libusb.org) is an open source project. It is a C library that gives applications easy access to USB device on many different operating systems. It provides a set of API interfaces for user to develop USB drivers. From the libusb source codes, we can see these APIs invoke the kernel’s underlying interface which are similar with the kernel driver technology. Because libusb is much closer to the USB specification, it is easier than kernel driver for developers. The detailed procedure is as follows:

1) Download the libusb source code;

2) Configure and make install on embedded linux system;
3) Copy the generated libs to ARM controller (eg: libusb-1.0.so, libusb-1.0.so.0, libusb-1.0.so.0.1.0)
4) Use the API functions to driver camera (eg: libusb_bulk_transfer / libusb_control_transfer to transfer image data or control commands)

2.2 *Camera image processing*

In industrial application, the camera is used to distinguish and locate the objects. Some image processing methods should be used to reduce noise or interference, like the gray conversion, binarization, smooth filtering, etc. After pre-processing, the key step is the template matching and object localization. The match method usually includes:

1) Least square error (*LSE*):

$$R(x,y)=\sum_{x',y'}[T(x',y')-I(x+x',y+y')]^2 \tag{1}$$

2) Correlation match:

$$R(x,y)=\sum_{x',y'}[T(x',y')\cdot I(x+x',y+y')] \tag{2}$$

3) Correlation coefficient:

$$R(x,y)=\sum_{x',y'}\{[T(x',y')-\bar{T}]\cdot[I(x+x',y+y')-\bar{I}]\} \tag{3}$$

Where, T is the template image and I is the camera image, $\bar{T}$ and $\bar{I}$ are mean values. *LSE* method searches the minimum value, correlation method uses the maximum value, and the coefficient finds the value closest to 1.

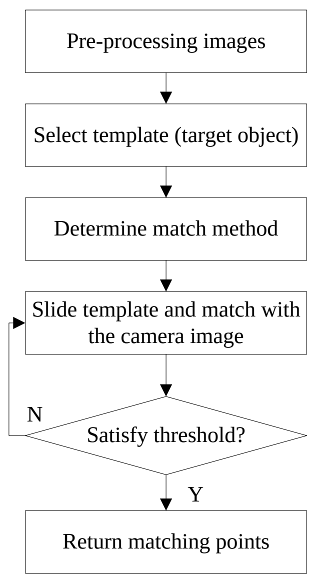

Figure 1. Camera image matching procedure

The detail procedure of image processing is shown in Figure 1. The match time has an important impact on system real-time. Two measurements are taken to reduce template matching time. One way is setting a threshold value to end the match searching; the other is scaling down the size of camera image and template image.

2.3 *Camera calibration and coordinate transformation*

Getting object position in the real world from a camera image, there is a serial of coordinate transformations. Seeing from the Figure2, it includes the image pixel coordinate (u,v) to image physical coordinate (X_f, Y_f), the image (X_f, Y_f)) to camera coordinate (x,y,z), the camera coordinate to the physical world coordinate (X_W,Y_W,Z_W). Based on the ideal pin-hole imaging model, we can get the transformation equations. From image pixel coordinate to the camera coordinate is:

$$\begin{cases} u-u_0 = fs_x x/z = f_x x/z \\ v-v_0 = fs_y y/z = f_y y/z \end{cases} \tag{4}$$

Image pixel to the world coordinate is:

$$z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix}\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = M_1M_2X = MX \tag{5}$$

Where, u,v are the image pixels, x,y are the value in camera coordinate. f_x、f_y、u_0、v_0 are the camera’s intrinsic parameters which relevant with camera internal structure. The translation vector T and rotation matrix R are extrinsic parameters.

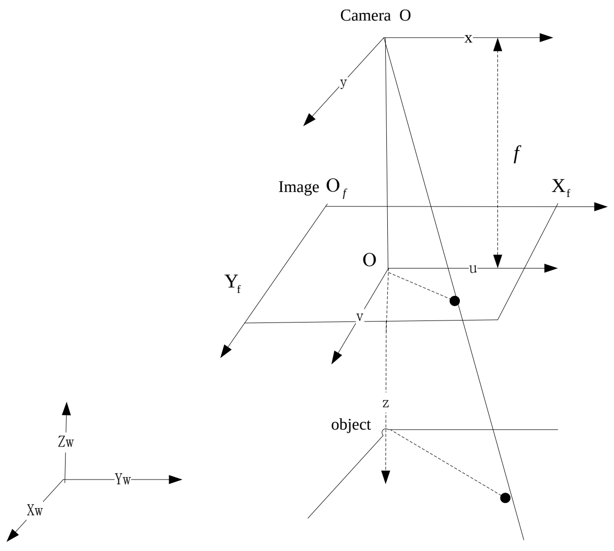

Figure 2. Coordinate transformation

The intrinsic parameters are determined by camera calibration. We use the Zhenyou Zhang’s method [4] and make a calibrate board. A set of images of the board are collected to calculate the parameters. Besides these 4 parameters, camera has radial distortion and tangential distortion for the lens‘ property and assembly errors. To improve calibration precision, the distortion parameters can be calculated using Brown's method[5]. But it will increase computational complexity, it is not considered here.

3 EXPERIMENTAL RESULTS

Hardware platform: OK210 controller, USB camera (1/3'' CMOS black and white, 36-megapixel, 54fps@752*480). The software: Embedded Linux system (2.6.35 version), Qt 4.8 for displaying image.

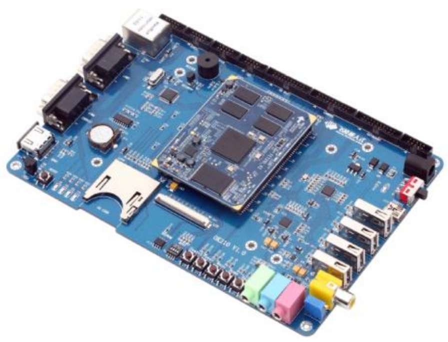

Figure 3. Hardware platform

3.1 *Image acquisition and processing*

The industrial camera is connected with ARM controller and captures images. In this part, it is used to identify different objects. We select the white triangle as template shape and calculate the least square error values. Figure 4(a) and Figure 4(b) are the match results. Blue box shows the match position. At Figure 4(a), the right-top triangle is found and the calculation ends. When some interference is generated at top triangle, it will find the bottom one which has the least square errors.

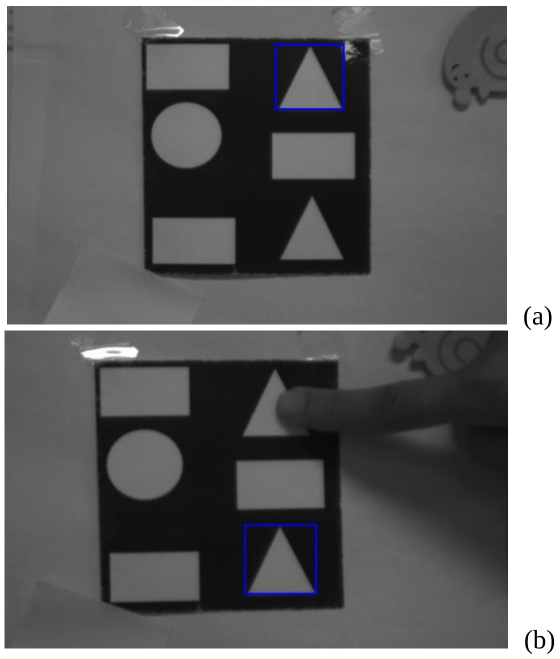

Figure 4. Template matching results

Here we apply OpenCV to process the images. The whole time of image processing and displaying on screen is nearly 2 seconds. The real-time performance should be improved further. Some measurements can be taken. For example, try parallel computing in OpenCV, optimize the data type transfer efficiency from OpenCV (IplImage) to Qt (QImage). Besides, the faster processor can be used to improve the processing speed.

3.2 *Camera calibration*

The calibration board made is white and black checkerboard. It has 5 lines and 6 columns with 30mm width. And 10 images are collected to calibrate camera. Figure 5 shows two among them. Figure 6 is the corner detection results.

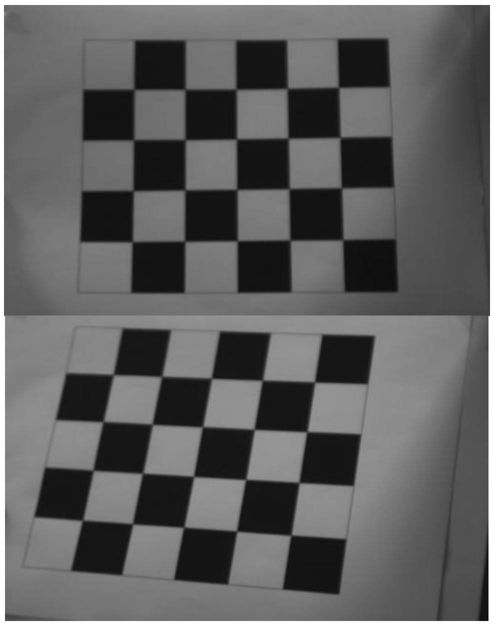

Figure 5.Two calibration board images

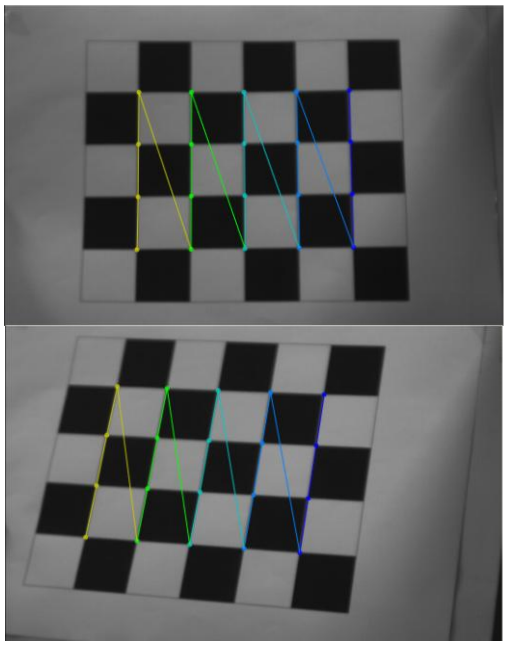

Figure 6. Corner detection

After calibration, the intrinsic parameters can be received, showed as equation (6). Extrinsic parameters are relevant with the actual position and each image has different extrinsic parameters.

$$\begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2133.2 & 0 & 518.0 \\ 0 & 2124.4 & 220.6 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

3.3 *Coordinate transformation*

The other image of calibration board is captured to test the calibration parameters, like Figure 7. Selecting 3 corners, such as the A, B, C, the acutal distance of AB and AC both are 30mm. Using calibratin parameters calculated above and the equation (4), the distance between A and B, A and C can be received. A(327,199), B(324,277), C(406,201) are the pixel positions in the image.

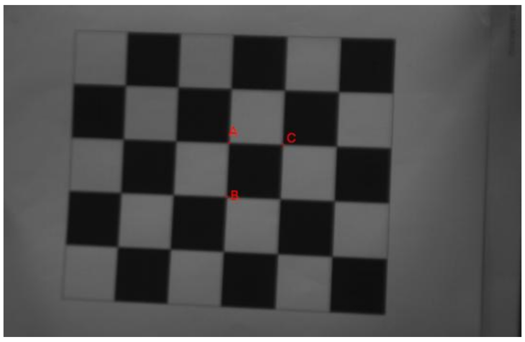

Figure 7. Image used to verify calibration parameters

Then, the distance is :

$$AB = z \cdot \sqrt{\left(\frac{u_A - u_B}{f_x}\right)^2 + \left(\frac{v_A - v_B}{f_y}\right)^2} = 29.0 \text{ mm}$$

$$AC = z \cdot \sqrt{\left(\frac{u_A - u_C}{f_x}\right)^2 + \left(\frac{v_A - v_C}{f_y}\right)^2} = 29.2\text{mm}$$

Where, z is the vertical distance between camera and object. Here the object is the calibration board and z is about 789mm. It can be seen the calibration parameter has little bias but basically meet the requirement. Considering the distortion parameters, it would be better.

Adding the extrinsic parameters (rotation matrix and translation vector), physical position of object in the world coordination can be calculated. Then, according to target position, the industrial robot will plan motion path to complete automatic carrying or assembly

4 CONCLUSIONS

The machine vision system based on embedded Linux is presented. For USB camera driver, the libusb is used to simplify development. The camera calibration and template matching are also introduced. Using calibration parameters and the matching position, the world coordination of target object can be calculated.

Accuracy and real-time are two key points for vision system on embedded Linux. From the experimental results, the calibration parameter is verified and the template matching is effective. They satisfy the accuracy requirement. The real-time of embedded processing system is important for industry application. Template matching is the most time-consuming part which should be optimized further. On one hand, the faster hardware platform can be used to improve processing speed. On the other hand, the parallel computing or software optimizing can be tried in the future.

ACKNOWLEDGEMENT

This work was financially supported by the Guangdong Introduced Leading Talents Project, the cooperation project of Guangdong Province and Chinese Academy of Sciences (2011B090300031), the cooperation project of Foshan and Chinese Academy of Sciences (2012HY100021).

REFERENCES

[1] Qian, Y. & Chen, S.L. 2013. Image acquisition and display with USB camera based on embedded platform. *Electronic Design Engineering* 21(3): 140-142.

[2] Huang, Z.F. & Chen, H.P. et al. 2012. Image processing of detection system for parts in industrial camera based on OpenCV and USB. *Modern Electronics Technique* 35(18): 128-132.

[3] Song, L.H. & Gao, K. 2010. Implementation of USB camera drive under Embedded Linux. *Computer Engineering* 36(9): 282-284.

[4] Zhang, Z. 2000. A flexible new technique for camera calibration. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 22(11): 1330-1334.

[5] Brown, D.C. 1971. Close-range camera calibration. *Photogrammetric Engineering* 37(8): 855-866.