Computer Vision Framework for Object Monitoring

Theerayod Wiangtong and Sethakarn Prongnuch
Electronics Engineering Department
Mahanakorn University of Technology
Bangkok, Thailand 10530
Email: theerayo@mut.ac.th

***Abstract*—This paper presents the use of Visual C++ integrated with OpenCV library as a framework for video analysis applications. Conventionally, applications are developed and control using textual-based Win32 console. In this work, however, the library can be applied in Windows form that helps users to visually control algorithms implemented in applications. The framework, written in OOP style, tailors the existing library into three main class methods; initializing, running and ending. Streaming data can be captured from USB cameras, IP cameras and video files. Implementing computer vision algorithms include object counting and object monitoring is demonstrated. Results show that algorithms work well and they can be developed in short time.**

I. INTRODUCTION

Building from scratch for image and video applications seems to be very hard and time consuming. Various image and video libraries have been developed and provided either commercially or academically. This paper focuses on an open source library named OpenCV which is written in C/C++ and able to run under Linux, Windows and Mac OS. Currently, there are over 500 functions [1] in many computer vision areas; including factory product inspection, medical imaging, security, user interface, camera calibration, stereo vision, and robotics, are designed for computational efficiency and with a strong focus on realtime applications. In OpenCV, these functions are natively used as Win32 console applications.

There are some efforts to use the library in Windows form. EmguCV and [2], for example, are the wrappers of OpenCV functions, written and used in C#. It allows developers to run algorithms using mouse clicking on buttons in Windows applications rather than using keyboard putting text in console applications. Additionally, developing on Object Oriented Programming (OOP), the framework can easily add new trial components for image and video processing.

The paper is organized as follows: section II describes related work. Section III provides a brief overview on the proposed framework. Section IV describes some experimental results. This work is then concluded in section V.

II. RELATED WORK

For open source licenses, VXL (the Vision-*something*-Libraries) written in ANSI/ISO C++ is a collection of functions for computer vision research and implementation. It can be portable over many platforms. CImg library is another example of open source C++ toolkit for image processing. It provides simple classes and functions to load save, process and display still images. Hence, it is not suitable for realtime application.

In terms of popularity, OpenCV dominates other open source libraries. It is well documented and gains many interests from developers. Additionally, [3] reported that OpenCV achieves higher performance to process image data, e.g. resizing, 2D DFT and optical flow, compared to VXL.

Commercially, MATLAB/SIMULINK Image Processing Toolbox™ provides a comprehensive set of reference-standard algorithms and graphical tools for image processing, analysis, visualization, and algorithm development. From academic work, Khoros [4], a visual environment developed at the University of New Mexico, is being used as a research and development tool in a variety of scientific applications, including geographical imaging systems (GIS), medical imaging, and distributed processing. In [5], a design and implementation of the innovative software development system for image processing and compression is presented.

EmguCV essentially provides an interface between OpenCV and C#. Although C# is more convenient than C/C++, the main drawback of using a wrapper written in C# is that OpenCV function names and data types are changed; i.e. using Emgu as a prefix. Also it is not free for commercial use.

To comfort users with original OpenCV notations, this work is based on C++. We introduce the framework constructed under objected oriented programming style. It focuses on an existing open source computer vision library and extends developing capabilities in Windows forms. Object monitoring algorithms tailored from OpenCV library are developed to count moving objects also detect abandoned/stolen objects in real-time.

978-1-4673-2025-2/12/$31.00 ©2012 IEEE

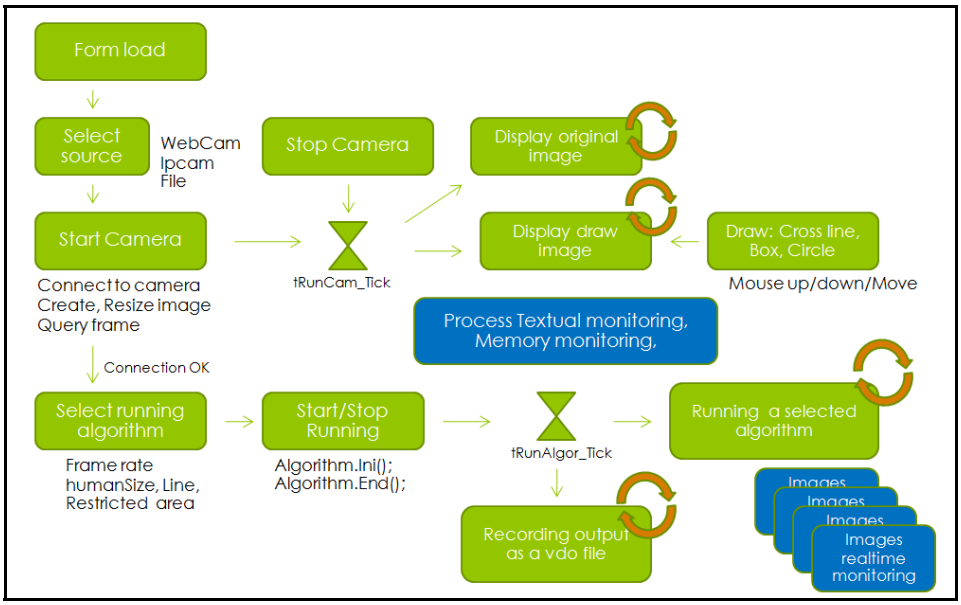


Figure 1. The proposed framework.

III. THE FRAMEWORK

The process inside the framework is depicted as a block diagram as shown in Fig. 1.

Steaming input data are captured using OpenCV functions, cvCaptureFromCAM and cvQueryFrame, resulting in IPL image format. This data format is regularly used in OpenCV functional processes. However, to display back in Windows form, all IPL images have to be converted using a method (bitmap) in Bitmap class.

Traditionally, OpenCV algorithms are built as functional programming and while loop command is used to capture images and run algorithms continuously. However in visual style using object oriented programming, while loop is not acceptable. Therefore, in this work, timer components (tRunCam_Tick and tRunAlgor_Tick) are exploited.

Algorithms or user-defined modules that contain several OpenCV functions are written as methods in a Class. Three main common methods introduced here are .ini(), .run() and .end(). The method .ini() contains cvCreateXXX functions (such as cvCreateImage, cvCreateMemStorage) to allocate heap memory as process buffers, as well as contains initial values of some parameters. The method .run() contains algorithms which are sequences of OpenCV functions or other user-defined modules. At last, the method .end() contains cvReleaseXXX functions (such as cvReleaseImage, cvReleaseMemStorage) must be called to return all resources back to the system, or memory leak problem may occur.

IV. ALGORITHMS

A. *Object Counting*

One of the widely-used features in surveillance systems is object tracking which aims to track and count vehicle [6] or people [7] in various conditions. This can be used for real-time traffic monitoring. Also the number of people counted in shopping center may be used for making some strategic economic decisions.

In this work, Blob tracking algorithm [8] shown in Fig. 2 is adopted. In brief, *Foreground/background detection module* is first used. It is followed by *Blob entering detection module* to detect new blob object entered to a scene on each frame. *Blob tracking module* is to track each new entered blob. *Trajectory generation module* collects all blobs positions and save blob's trajectory to hard disk. *Trajectory postprocessing module* optionally performs a blob trajectory smoothing function.

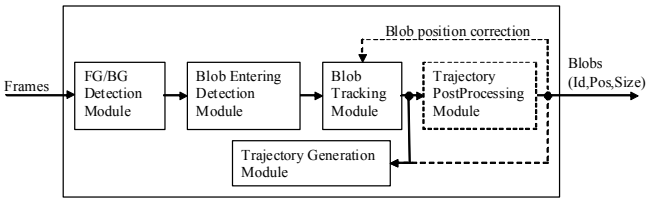


Figure 2. Blob tracking modules.

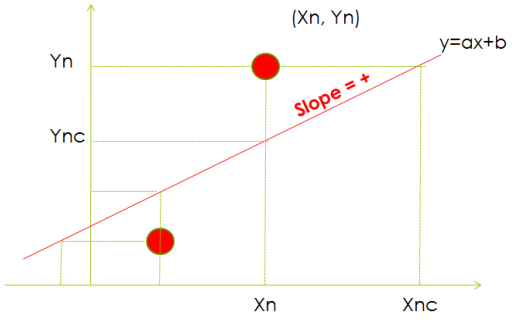


Figure 3. Point of tracking object *(Xn, Yn)* compared to the crossing line.

Thanks to object oriented programming, these function modules are easily integrated as a new class and used in this

framework. Predefined tracking mechanisms such as connected component, mean-shift, particle filter or Kalman filter can be selected using dropdown list. Our preliminary tests show that connected component tracking method works well in this system compared to the others.

Tracking information of several moving objects can be exploited to count the number of cars or people passing a line or area. In this work, line or area must be predefined as a condition to count objects when they are crossing. This framework allows users to draw a straight or freehand line or circle directly into the video frame using their mouse pointer which is very comfortable and effective. After a straight line is determined, two parallel lines are added over and under this line, the counter increases when a tracked object crosses both lines. Moving direction (in or out) is then defined.

Fig. 3 shows how to determine tracking point status (above or below) compared to the line. When the line slope is positive, the following condition in (1) is used.

$$Status = \begin{cases} 1 & above\ if\ Xnc > Xn,\ Ync < Yn \\ -1 & below\ if\ Xnc < Xn,\ Ync > Yn \\ 0 & edge\ \ if\ Xnc = Xn,\ Ync = Yn \end{cases} \quad (1)$$

In case the slope is negative, the conditions are changed to

$$Status = \begin{cases} 1 & above\ if\ Xnc < Xn,\ Ync < Yn \\ -1 & below\ if\ Xnc > Xn,\ Ync > Yn \\ 0 & edge\ \ if\ Xnc = Xn,\ Ync = Yn \end{cases} \quad (2)$$

Hence the passing direction is obtained from:

$$Obj_{Direction} = Obj_{Status(t)} - Obj_{Status(t-1)} \quad (3)$$

For example, if $Obj_{Direction}$ equals 2, the object comes across the line from below and vise versa.

B. *Object abandoned/stolen detecting*

The main function utilized for this purpose is background/foreground segmentation. The background is updated from time to time. This static information is compared with incoming frames. When there is no movement or activities, it can intelligently alert whether static objects in frames are missing (stolen) or adding in (abandoned). The alert happens for a while before the static background is updated.

V. Implementation and Experiments

OpenCV library version 2.2 is downloaded and utilized in Visual Studio 2008 to develop algorithms in computer vision system. From Fig. 4, function group boxes for various control objectives are shown and described as follows.

- *Input selection:* Multi-types of streaming video input are supported, i.e. USB camera, IP camera and video file. In this work, Vivotek IP camera and Creative WebCam are used.
- *Image size:* Due to different resolutions from different sources, we have to predefine the size of input images those will be processed in this system. Standard format such as CIF, QCIF, 4CIF are provided.
- *Algorithm selection:* An algorithm under test will be selected using radio buttons or drop down lists in this area.
- *Adjust parameters:* In different light or environment conditions, there are parameters needed to be adjusted for the best results. Instead of changing in textual command, slide bars and text boxes are used. Developers can see effects in realtime.
- *Frame rate*: To accelerate or slow the process time, input frame rate can be changed as appropriate.

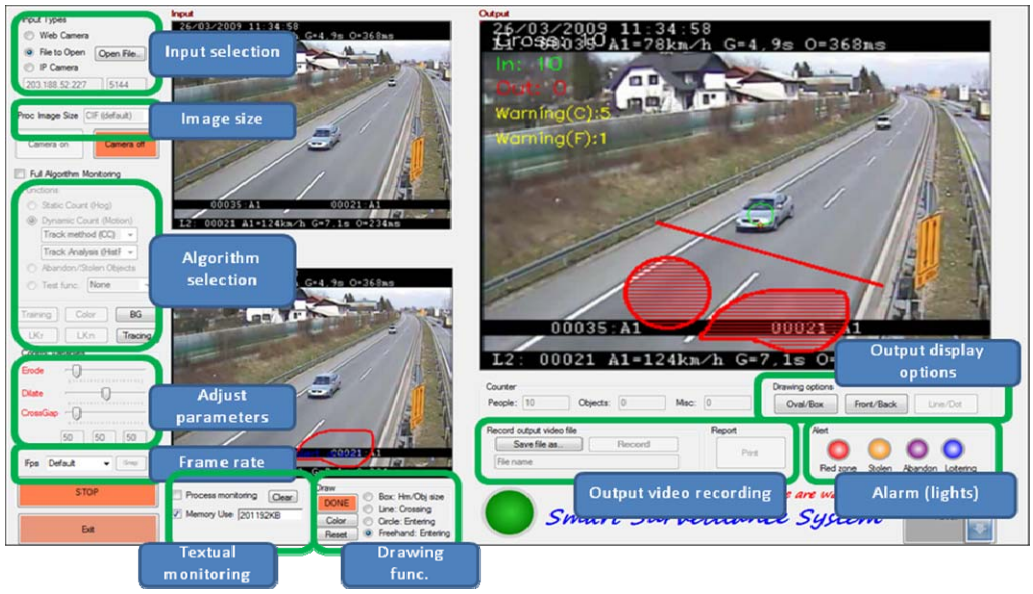
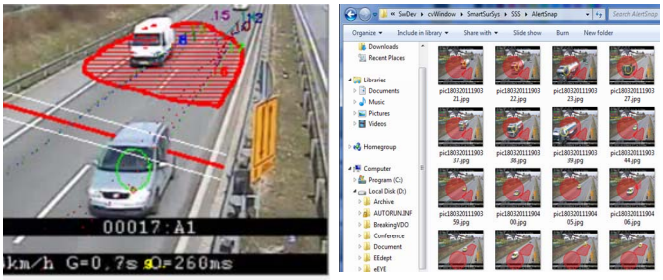


Figure 4. User interface.

- *Textual monitoring:* To monitor an ongoing process such as camera connection time, image size, running algorithm, alerts, memory use etc., a text box underneath the low-left picture can be toggled into the front to display system information.
- *Drawing functions:* Developers can easily draw a line, circle, or freehand directly into the image. For example, in object counting algorithm, we may have to define a crossing line, as well as restricted or alert area. The size of interest objects can be bounded using rectangular box to guide some tracking mechanism.
- *Output video recording:* Streaming output can be recorded as a video file in different format.
- *Alarms:* These virtual LEDs also the synthesized sounds can be used to notify developers. Additionally, snapshots will be taken and kept into hard disk.
- *Output display options:* These buttons are added to enable or disable results such as tracking line, texts on image.



a) Car tracking and snapshots when objects appear in restricted area.

b) Object (bin) is moved.

Figure 5. Experimental results.

Results of the algorithms, to process video input at 15 frames per second, CIF-size (352 × 288), are demonstrated in Fig.5. From the tracking algorithm, Fig.5 a) shows tracking marks of moving objects, cars in this case, in different colors. On average, counting accuracy for this highway traffic scenario is 94%. Since, from this camera angle, two adjacent cars having the same speed can cause occlusion and counting error. The right side shows snapshots taken when objects appear in the restricted (red) area. However, if the size of tracking objects (a car in this case) is predefined and used as an additional condition, the accuracy can be improved by a few percent.

Fig. 5 b) demonstrates the functionality of detecting abandoned/stolen objects. After a missing object (a bin) has been moved by a man, a red box is shown in the scene.

VI. CONCLUSIONS

This work introduces computer vision developing structure that combines the computer vision library and extends developing capabilities in Windows forms. The development is based on objected oriented programming, written in C++. Three main class methods introduced in this development framework includes .ini() to initialize image and data buffers, .run() to run a designed algorithm and .end() to destroy current objects and return resources to the system.

The object monitoring algorithms are demonstrated how to integrate video processing functions in this framework. Users can exploit their mouse to change functionalities or algorithm parameters in Windows form. Results show that the algorithms can be used to count moving objects also detect stolen/abandon objects in real-time.

ACKNOWLEDGMENT

This work is partially supported by Infinite Electric Company for equipment and funding.

REFERENCES

[1] Gary Bradski and Adrian Kaehler, Learning OpenCV, 1st Edition, O'Reilly Media, Inc., 2008.

[2] A. Nagy, Z. Vamossy, "OpenCV C# wrapper based video enhancement using different optical flow methods in the super-resolution", International Symposium on Intelligent Systems and Informatics, 2008, pp. 1 - 6 .

[3] Utkarsh Sinha (2010, July 10), OpenCV vs VXL vs LTI: Performance Test [Online]. Available: http://www.aishack.in/2010/07/opencv-vs-vxl-vs-lti-performance-test/

[4] Konstantinos Konstantinides and John R. Rasure, "The Khoros software development environment for image and signal processing", IEEE Transactions on Image Processing, May 1994 Volume: 3 Issue: 3, pp. 243 – 252.

[5] Cyganek Bogusław and Borgosz Jan, "An Object-Oriented Software Platform for Examination of Algorithms for Image Processing and Compression", International Conference on Computational Science, 2003, pp. 713-720.

[6] Thou-Ho (Chao-Ho) Chen, Yu-Feng Lin, and Tsong-Yi Chen, "Intelligent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance", International Conference on Innovative Computing, Information and Control,2007, pp. 238 – 238.

[7] Donatello Conte, Pasquale Foggia, Gennaro Percannella, Francesco Tufano, and Mario Vento, "A Method for CountingMoving People in Video Surveillance Videos", EURASIP Journal on Advances in Signal Processing, Volume 2010, pp. 1 – 11.

[8] OpenCVWiki, Video Surveillance / Blob Tracker Facility [Online]. Available: http://opencv.willowgarage.com/wiki/VideoSurveillance