A General Framework for Tracking Objects in a Multi-Camera Environment

Karlene Nguyen, Gavin Yeung, Soheil Ghiasi, Majid Sarrafzadeh
{karlene, gavin, soheil, majid}@cs.ucla.edu

Abstract

We present a framework for real-time tracking of objects. Our system consists of multiple cameras and a control unit that communicate through a network. Each camera has a general-purpose processor and a reconfigurable hardware unit embedded in it. Therefore, some computation can be performed at the point of data collection. We argue that collocating the computation with the data at vision sensors can improve performance, communication overhead and network scalability. We exploit the parameterization and tuning of the vision algorithms and present a sample tracking application implemented on our framework. We further argue that the proposed architecture can be used to implement many other real-time vision applications through hardware reconfiguration.

1. Introduction

There is a recent trend towards object tracking applications, such as surveillance or traffic control and management. Such applications require the use of unsupervised detection of events. By utilizing vision sensors, massive amounts of image data can be collected and hence, used by image-processing algorithms to enhance the knowledge of the event without supervision.

This data must be processed quickly due to the dynamic real-time nature of the input in order to generate quick responses. In order to generate quick responses, the intensive and dynamic image-processing algorithms used must be able to load and run quickly. While software realization of such algorithms simplifies the user task to implement them, it tends to run slower than its hardware counterpart.

Furthermore, many applications require the vision algorithms to be dynamically tuned and parameterized. This tuning is usually performed based on the current information available from the scene. Therefore, a non-flexible hardware implementation of vision algorithms cannot be used for such applications. Implementing these algorithms on reconfigurable hardware fabric, such as FPGA, can continue to allow flexibility while running an algorithm much faster than the corresponding software implementation.

Traditionally, vision sensors collecting the image data are located separately from computational resources that process the image data. To circumvent the computational load of vision algorithms, data processing can be combined with the vision sensor into a single module. In our project, we use a video camera that has both an FPGA and a general-purpose processor embedded in it. These resources can be used for such computations.

In this project, we utilize multiple video cameras, each with an embedded general-purpose processor and an FPGA. By collocating the computations with the image data on a given camera, the network scalability and communication overhead are significantly improved. Moreover, each computation can be mapped to different resources of the camera based on performance requirements, energy dissipation, development cost and other metrics.

We create an initial framework for tracking objects in a multi-camera environment. We present a sample application and an algorithm for selecting the proper image processing algorithm and computational resource. Then, we present a general framework for tracking objects with multiple video cameras. We further argue that our application, which allows us to track objects in an environment, would be ideal for implementation on an FPGA, in which reconfigurability could introduce a significant speedup. In the future, we hope to contrast the speedup of the implementation in software and FPGA and create an application in which both resources can be dynamically utilized to generate maximal performance.

2. System Setup

Our platform is composed of two IQEye3 video cameras provided by IQInvision [1]. Each camera has 16 MB memory, a 250 MIPS PowerPC, and a Xilinx Virtex 1000E FPGA embedded in it. A limited number of libraries allow developers to generate custom "C" applications for the cameras. The FPGA can also run an algorithm on the streamed real-time image data. In future work, we hope to exploit the advantages of the FPGA reconfigurable fabric since repetitive computationally intensive image processing can be loaded to the FPGA for faster processing. Full access to the raw image allows an unlimited possibility of applications. The IQEye3 cameras run a simplified version of the Linux operating system with full networking functionality. Each camera can communicate via an Ethernet connection and/or serial port. Hence, the camera can be configured to send images via e-mail and FTP as well as send data using standard socket programming. The camera also serves dynamic web pages, allowing remote users to view live images via a web browser from the camera.

A main controller on a desktop computer is used to retain the current state of the scene. Using an algorithm, the main controller determines which image-processing algorithm should be selected and loaded onto a given IQEye3 camera as well as the parameters for the specified algorithm. Once the specified algorithm on the camera terminates, the numerical result(s) of the algorithm is returned to the main controller in order to update the current state of the scene. The controller and IQEye3 cameras use a TCP/IP based messaging system to communicate. Figure 1 shows the general framework of the described setup.

Each time an image-processing algorithm completes, a processed image is transferred to an FTP server. A webpage has been developed to dynamically display the contents of the processed images located on the FTP server. This webpage is reloaded automatically every one second.

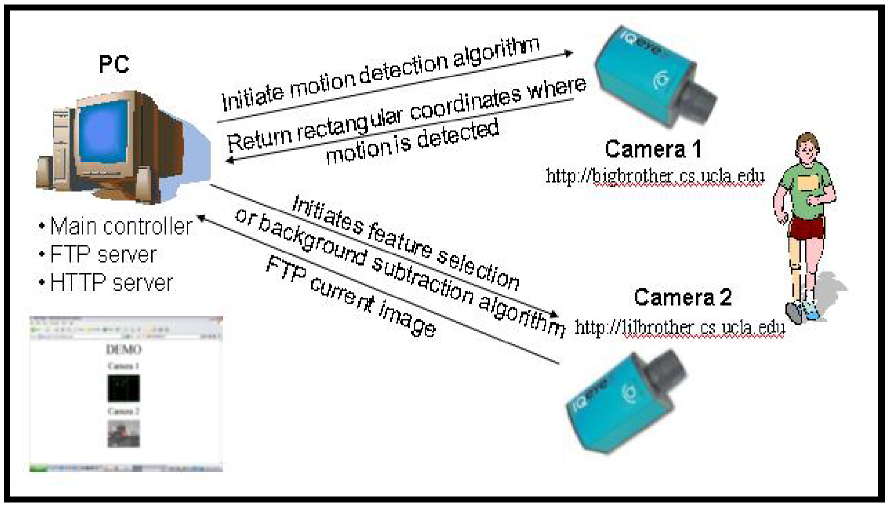
PC
Initiate motion detection algorithm
Return rectangular coordinates where motion is detected
Camera 1
http://bigbrother.cs.ucla.edu
• Main controller
• FTP server
• HTTP server
Initiates feature selection or background subtraction algorithm
FTP current image
Camera 2
http://lilbrother.cs.ucla.edu
DEMO

Figure 1. Cameras collaborating with the control unit to track an object.

3. A Sample Application

We have implemented three well-known vision algorithms on the camera platform. These algorithms are all widely used in surveillance and tracking applications. They include: motion detection, background subtraction and feature selection. The motion detection algorithm subtracts the pixels of two successive frames in order to determine whether there is motion between those frames. The background subtraction algorithm subtracts an initial frame containing the background information from another frame in order to determine if there exists a foreground object against the background. The feature selection algorithm chooses points that are deemed interesting in a given frame for future use in conjunction with a feature-tracking algorithm.

We have developed a simple controller, which tries to track the motions in a specific area of interest. By combining these algorithms in a useful manner, we created a simple application that allows the main controller to learn more information about the current state of the scene. Therefore, it can dynamically respond to a given input. Moreover, this application can exploit the reconfigurable hardware fabric existing in the system framework.

Figure 1 shows an overview of the general architecture of the application. The cameras are oriented coaxially in order to perceive the identical scene. As an object moves into the field of view of the cameras, the main controller is notified by returning particular values. The main controller updates its knowledge about the particular scene (i.e., motion, foreground object, etc.) and may send a message to particular cameras in order to initiate particular algorithms with specified parameters. As each algorithm runs, the resulting image is sent to an FTP server.

Figure 2 shows a hierarchical diagram of the application. Each node represents an image-processing algorithm. Based on the output of the algorithm, the controller can update the current information about the state of the scene and dynamically specify a particular image-processing algorithm to run on another camera. Further, various parameters can be dynamically specified for the feature selection algorithm based on current information available from the scene. For example, specifying the rectangular coordinates of the region where motion is detected lessens the amount of computation for selection the features.

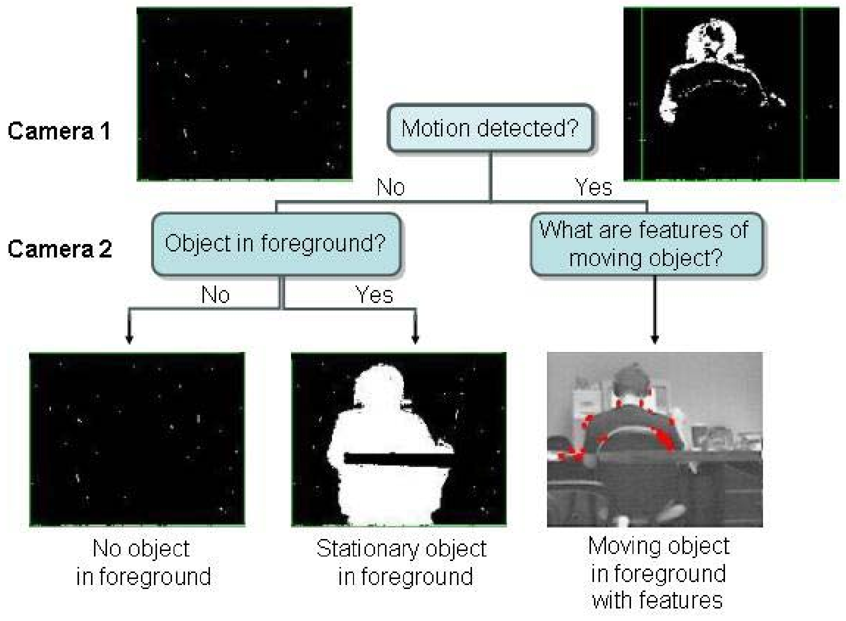
Camera 1
Motion detected?
No
Yes
Camera 2
Object in foreground?
What are features of moving object?
No
Yes
No object in foreground
Stationary object in foreground
Moving object in foreground with features

Figure 2. Overview of a sample application

It is assumed that one of three following scenarios can occur in a given scene at any given instant:

1. There is no foreground object, which implies that there is no moving object
2. There is a foreground object and the object is not moving
3. There is a foreground object and the object is moving

In the case of scenario 3 in which a moving object is present, a feature selection algorithm runs on the region of the image in which the moving object is present. Determining the features of an object is useful in tracking the direction, velocity, and other characteristics of the object. Restricting the feature selection to the moving object's region improves the performance, since most of the applications are not interested in tracking the still objects existing in the background.

Our application distinguishes between these three scenarios and supplies the main controller with the most amount of information about the particular scenario (i.e., the feature coordinates of a moving object). According to Figure 2, camera 1 runs the motion detection algorithm repeatedly, while camera 2 would oscillate between running background subtraction and feature selection based on the output of camera 1.

Each time the motion detection algorithm runs on camera 1, it returns the resulting value to the main controller. If there is motion, the rectangular coordinates of the region where motion occurs is passed to the feature selection algorithm. If there is no motion, the background subtraction algorithm is initiated on camera 2. In case of hardware realization of the algorithms on the FPGA, complete reconfiguration would only be necessary each time the algorithm is switched. If input parameters to an algorithm residing in the FPGA changes, it can be reconfigured using parameterization. Parameterization is performed much faster than the complete reconfiguration and does not have a significant reconfiguration delay overhead.

After each of the algorithms run on a camera, the resulting image is sent via FTP to the PC, which hosts a web server. A webpage is refreshed every one second in order to dynamically display the resulting image on each camera. Figure 3 shows a snapshot of the dynamic webpage.

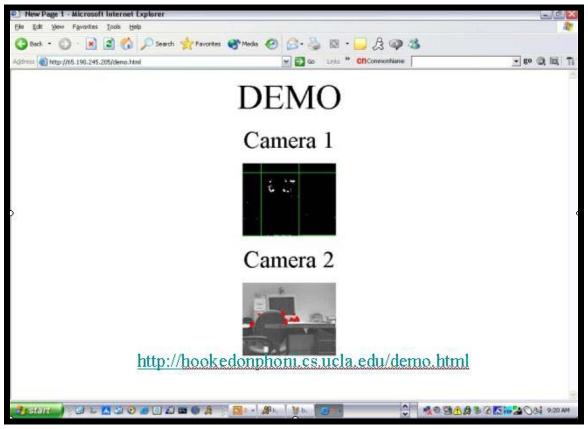
New Page 1 - Microsoft Internet Explorer
DEMO
Camera 1
Camera 2
http://hookedonphoni.cs.ucla.edu/demo.html

Figure 3. Dynamic webpage displays real-time images from two IQEye3 cameras collaborating the in framework

4. Conclusion and Future Directions

In this paper, we present a framework for real-time tracking of objects. This system consists of multiple Iqeye3 cameras and a control unit that communicate using a messaging mechanism over the network. A simple tracking application, which requires collaboration of cameras and control unit, is implemented using this framework.

Despite our pure software implementation of the vision algorithms on the vision sensors, the embedded reconfigurable hardware fabric can also be utilized for realizing algorithms. Implementing algorithms on this computation resource has different performance, development cost and energy dissipation characteristics compared to the software counterpart. Therefore, an intelligent choice of the target architecture for each of the algorithms can improve the system performance drastically. We plan to extend our work to include the image-processing algorithms implemented on the FPGA and exploit hardware reconfiguration in vision applications.

Several other extensions of this project include adding more algorithms to the database of algorithms, a more realistic and useful application of the image-processing algorithms, an actuation unit to allow digital pan-and-tilt capabilities for feature tracking, and the ability to track multiple objects in a scene.

5. References

[1] IQinVision Online Documentations, IQinVision Inc., http://www.iqinvision.com.

[2] Carlo Tomasi and Takeo Kanade, "Detection and Tracking of Point Features", *Carnegie Mellon University Technical Report CMU-CS-91-132*, April 1991.

[3] A. Benedetti, P. Perona, "Real-time 2-D Feature Detection on a Reconfigurable Computer", *in IEEE Conference on Computer Vision and Pattern Recognition*, June 1998, Santa Barbara, CA.

[4] A. Bissacco, A. Chiuso, Y. Ma and S. Soatto, "Recognition of human gaits**"**, *In Proc. of the IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, vol II, pages 52-58, 2001.

[5] X. Feng, P. Perona, "Real Time Motion Detection System and Scene Segmentation", *CDS TR CDS98-004*, Caltech, 1998

[6] H. Jin, P. Favaro and S. Soatto, "Real-time Feature Tracking and Outlier Rejection with Changes in Illumination", *Proc. of the Intl. Conf. on Computer Vision*, July 2001.

[7] P. Athanas and L. Abbott, "Addressing the Computational Requirements of Image Processing with a Custom Computing Machine: An Overview", *in Proceedings of the 2nd Workshop on Reconfigurable Architectures*, April 1995, Santa Barbara, CA.

[8] S. Smith and J. Brady, "Asset-2: Real-time Motion Segmentation and Shape Tracking", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, no. 17, pp. 814-820, 1995.

[9] Xilinx Online Documentation, Xilinx Inc. www.xilinx.com.