Implementing a Real-Time Application on a TMS320C40 Multi-DSP

Authors: F. Garzulin, B. Micovicova, N. Moulin, O. Seghrouchni

EFRIE, France

December 1995
SPRA312

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Figures

Implementing a Real-Time Application on a TMS320C40 Multi-DSP

Abstract

This application report describes a method for real-time processing of images captured using a CCD (charge-coupled device) video camera using the Texas Instruments (TI™) TMS320C4x digital signal processor (DSP).

The TI TMS320C4x is one of five generations of digital signal processors in the TI TMS320 family. The parallel processing ability of the TI TMS320C4x supplies the necessary performance for the computations in this study.

The SynDEx programming environment takes advantage of the capabilities of the TMS320C4x and monitors its time behavior.

Wavelet transform theory is used for convolution of the input image with the system response. The algorithm includes a method for edge detection and determining local maxima.

Hardware used in this study includes a PC, a CCD video camera, and the HEPC2-M card serving as the motherboard for up to four TI modules. The card is configured with one HET40SDX module and one HETVIO module.

This document was an entry in the 1995 DSP Solutions Challenge, an annual contest organized by TI to encourage students from around the world to find innovative ways to use DSPs. For more information on the TI DSP Solutions Challenge, see TI's World Wide Web site at www.ti.com.

Product Support on the World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Material Overview

TMS320C40 DSP

The TI TMS320C4x generation of 32 bit processors is designed specifically to meet the needs of parallel processing and other real-time embedded applications.

History

The TMS320C4x is one of five generations in the TMS320 family of digital signal processors. The TMS320C1x, TMS320C2x, and TMS320C5x offer designers a complete line of general-purpose and application-specific 16 bit DSPs. The TMS320C3x and TMS320C4x generations round out the TMS320 family, providing an ensemble of 32 bit DSPs.

The TMS320 family has grown from a single device, theTMS32010, introduced in 1982 to nearly 30 different products across five CPU architectures. On-chip hardware multipliers, register files, barrel shifters, ALUs, ROM, RAM, caches, and I/O peripherals along with massive internal bussing (all within a product as programmable as a general purpose microprocessor), make TI TMS320 devices well-suited for a wild range of computer-intensive applications.

Features

The TMS320C40 generation comprises parallel-processing devices and parallel-processing development tools. The following features make the TMS320C40 suited to applications requiring large amount of computations:

- High performance DSP CPU
- Six channel DMA coprocessor
- Two identical high data rate single-cycle transfer buses supporting shared memory systems
- Six communication ports for inter-processor communication

Material Description

We used the following hardware equipment to design the software developed for this project:

- HEPC2-M card
- PC486
- Super VGA monitor
- CCD video camera LDH 0470/00

The HEPC2-M is a PC-AT plug-in card that serves as a motherboard for up to four TI modules (TIMS) and provides a choice of interfaces between the PC bus and TIM-40s. Our HEPC2-M configuration consists of one HET40SDX module and one HETVIO module.

About Mathematical Wavelet Transform

The term 2D smoothing function describes any function $\theta_s(x,y)$ whose integral over x and y is equal to 1 and converges to 0 at infinity.

The image $f(x,y)$ is smoothed at different scales s by a convolution with $\theta_s(x,y)$. We then compute the gradient vector

$\nabla(f*\theta_s)(x,y)$

The direction of the gradient vector of a point (x_0,y_0) indicates the direction in the image plane (x,y) along which the directional derivative of $f(x,y)$ has the largest absolute value. Edges are defined as points (x_0,y_0) where the modulus of the gradient vector is greatest in the direction toward which the gradient vector points in the image plane. Edge points are inflection points of the surface $f*\theta_s(x,y)$.

We define two wavelet functions $\Psi_s^1(x,y)$ and $\Psi_s^2(x,y)$ such that:

$$\Psi^1(x,y) = \delta\theta_s(x,y)/\delta x \qquad \Psi^2(x,y) = \delta\theta_s(x,y)/\delta y$$

Let $\Psi_s^1(x,y) = \frac{1}{s^2}\Psi^1\left(\frac{x}{s},\frac{y}{s}\right) \qquad \Psi_s^2(x,y) = \frac{1}{s^2}\Psi^2\left(\frac{x}{s},\frac{y}{s}\right)$

The wavelet transform of $f(x,y)$ at the scale s, computed with respect to the wavelet $\Psi^1(x,y)$ is defined by:

$$W_s^1 f(x,y) = f * \Psi_s^1(x,y)$$

The wavelet transform of $f(x)$ with respect to $\Psi^2(x)$ is:

$$W_s^2 f(x,y) = f * \Psi_s^1(x,y)$$

One can prove that $\left(W_s^1 f(x,y), W_s^2 f(x,y)\right) = s\nabla\left(f * \theta_s\right)(x,y)$

Hence, edge points can be located from the two components $W_s^1 f(x,y)$ and $W_s^2 f(x,y)$ of the wavelet transform.

Let $$M_s f(x,y) = \left(\left(W_s^1 f(x,y)\right)^2 + \left(W_s^2 f(x,y)\right)^2\right)^{1/2}$$

$$A_s f(x,y) = \text{argument}\left(\left(W_s^1 f(x,y)\right)^2 + \left(W_s^2 f(x,y)\right)^2\right)$$

As in the Canny algorithm, the sharp variation points of $f*\theta_s(x,y)$, are the points (x,y) where the modulus $M_s f(x,y)$ has a local maxima in the direction of the gradient given by $A_s f(x,y)$. We record the position of each of these modulus maxima as well the values of the modulus $M_s f(x,y)$ and the angle $A_s f(x,y)$ at the corresponding locations.

C-Algorithm

Two methods are available to create a 2D Wavelet Transform:

- Quinconce
- Kernel separation

The Quinconce method uses the same dimension for filter and image (2D in our case). But the number of operations needed is too great. Therefore, we prefer the kernel separation method, which utilizesas many 1D wavelet transforms as signal dimensions.

For an image, 1D wavelet transforms are made on rows (W1) and on columns (W2).

Starting from the original digitized image, the program achieves four manipulated images by using two one-dimensional convolutions with the wavelet filters, by calculating modulus and phase of every pixels, and by extracting the maxima for edge detection.

Capturing the Image

The image seen by the camera is captured via the `CAPTURE_SINGLE( )` function in the VRAM of the HETVIO module, in a `512*512` format. The image is saved in the VRAM framestore0.

Wavelet Transform

We process a convolution in vertedge and in hortedge direction from the image *(x,y)* coordinates. The kernel of the convolution filter used is given by the wavelet theory. It assumes that the coefficients are:
a0 = -2, a1 = 2.

A line or a column of the image array (*NxN*) can be represented by a vector such as:

$$X(x_0,....,x_n,....,x_N)$$

The result of the convolution of this vector by the Canny filter is given by the following expression:

$$y_n = \sum_k x_{n-k} a_k$$

$$y_n = 2x_n - 2x_{n-1}$$

$Y(y_0,....,y_n,....,y_N)$ is a line or a column of the resulting image array (*NxN*) alter convolution.

So the vertedge convolution will detect the high vertical image data frequencies (image W1) and the hortedge one will provide the same effect on the horizontal direction (image W2). Using a symmetric approach concerning the first row and the first column of the image solves Border effects.

Wavelet-Modulus

W1 and W2 are used to calculate the modulus image according to the equation:

$$M_s f(x,y) = \left(\left(W_s^1 f(x,y)\right)^2 + \left(W_s^2 f(x,y)\right)^2 \right)^{1/2}$$

Wavelet-Angle

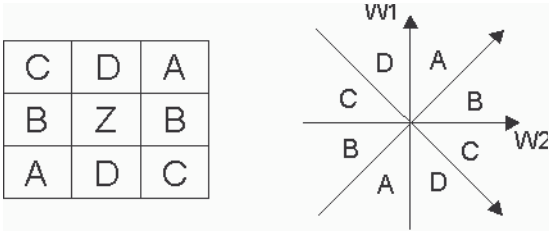
Same as Modulus, but calculating the angle of the gradient vector, according to:

$$A_s f(x,y) = \arctan\left(W_s^2 f(x,y) / W_s^1 f(x,y)\right)$$

Local Maxima Research

This function estimates the local maxima of an image. On one hand, it sets the local maxima point to 255 (white pixel value), on the other hand the value is set to zero (black pixel value). As a result, this function provides a binary image. Modulus and angle of the vector defined by W1 and W2 are computed by the local maxima research function. This function is able to distinguish four different directions.

Figure 1. Vector Defined by W1 and W2



As a first step, the appropriate neighbor pixels (A, B, C or D) are chosen, depending on the wavelet angle. A comparison with the central pixel follows, which will be marked as a maximum if it is greater than the two neighbors chosen.

Local maxima ensures that edges are represented by only one pixel and the image is in a binary mode (black or white).

Remark

The application result on two similar following images captured by our camera does not give the same edge-pixel position. It is a real problem if we want to do moving detection. Indeed, the difference of two similar following images should give a black image. But as two identical objects are not represented by the same pixels in two consecutive images, the difference cannot represent moving detection.

Algorithm Parallelization

Parallel computing systems provide high performance computing and are generally used to accelerate numerical computer applications. The SynDEx programming environment presents a methodology used to take advantage of the computation power of the system in a case of a real-time application.

Parallelism Methodology

With this method, the application algorithm as well as our edge detection algorithm are specified with graphs. Then the implementation of the algorithm on our hardware architecture composed by the HETVIO and the TIM modules with respect to real-time constraints may be formalized in terms of graph transformations. This allows us to optimize the real-time performances of the implementation, taking into account critical inter-processor communications.

As a result, real-time distributed executives are produced automatically without deadlock and with minimum overhead. This drastically reduces the development cycle of real-time applications.

Each typical application is composed of a computer-based system and of the environment with which it interacts. The system itself is composed of a hardware part executing a software part.

Finally the software part includes an application program, coding an algorithm independently of hardware consideration and executing hardware resources to execute the application program. The system is said to be reactive because, to keep control over its environment, it must interact permanently with it.

Implementation on a Multi-Processor with SynDEx Software

SynDEx has a graphical user interface which allows the user to input both hardware and software graphs. Then the user executes the SynDEx heuristics, which establishes a distribution and a schedule minimizing the response time.

As a result, the predicted time behavior of the algorithm on the multiprocessor is visualized. This feature is very important because the user can evaluate real-time applications even before the hardware is available. It is sufficient to know enough information about the application to experiment with its real-time capabilities for a given algorithm. Moreover, by using the performance prediction diagram, we can iterate at this level, trying different hardware solutions to size the architecture.

With the software we have obtained the following results:

```
with:  optimal path :              14.04 sec.
   monoprocessor architecture :    18.34 sec.
   biprocessor architecture :      16.20 sec.
   (including communication actions)
```

This distribution/scheduling allows us to have our algorithm executed in 16.20 seconds instead of 18.34 seconds on monoprocessor architecture.

Now, we have to verify these results by implementing the algorithm on a biprocessor architecture and calculating the time duration to compare it with the duration of the algorithm on a monoprocessor architecture.