

# COMPUTE-RESOURCE ALLOCATION FOR MOTION ESTIMATION IN REAL-TIME VIDEO COMPRESSION

Joseph Yeh and John Wawrzyniek

University of California at Berkeley, Department of EECS, Berkeley, CA 94720  
jyeh,johnw@eecs.berkeley.edu

## ABSTRACT

We present a novel method for allocating computational resources in motion estimation for real time video compression, in order to attempt to minimize prediction error over the whole image. The method makes decisions about how much compute power is allocated to motion estimation for each particular block based upon past observations of the input and output data of each task. We present preliminary results that demonstrate the ability of our method to improve the resulting quality of encoded video under a real-time computational constraint.

## 1. INTRODUCTION

Although the performance of computing devices continues to grow, new multimedia applications still arise to challenge the capabilities of these devices, particularly media compression algorithms such as H.264. At the same time, recent technological and market trends are pushing devices to handle more computational tasks while consuming minimal power. Many of these tasks have real time constraints, and therefore cannot be delayed. We present a framework in which such tasks can be performed at different levels of quality, with higher levels of quality requiring more computational resources in terms of percentage of CPU time in scalar architectures and/or functional units in highly parallel architectures, where the levels are set by a evaluator which takes into account the computational constraint of the system, and the characteristics of the input and output signals.

This framework needs to be robust to changes in the nature of the input signals and the overall computational constraints. Therefore, it does not rely on any *a priori* estimates of quality for a given implementation. Instead, it relies on observations of task inputs and task outputs to both establish the current tradeoffs between computational costs and quality and allocate computational resources among different tasks appropriately. Furthermore, the framework itself is computationally efficient.

The next section describes the mathematical framework and terminology used in our model. Although the framework has been previously presented in [1], it is repeated

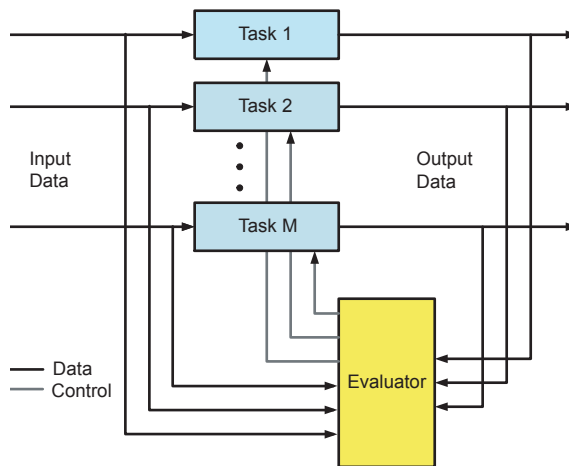


Fig. 1. System Diagram

here for clarity. Section 3 then explains how the model is applied in motion estimation and Section 4 presents some preliminary results. Section 5 presents possibilities for future directions and concludes the paper.

## 2. FRAMEWORK

The basic block diagram of our framework is shown in Figure 1. The computational device is running  $M$  tasks simultaneously. Each of the tasks can be run at a certain *implementation level*, and each implementation level of each task has a *computational cost* of  $c_m(l)$ , where  $m$  is the task number and  $l$  is a non-negative integer corresponding to the implementation level of the task. The tasks are ordered so that  $c_m(l)$  increases with  $l$ . In a particular time period, the total cost  $\sum_{i=1}^M c_m(l)$  of the selected implementations of all tasks cannot exceed  $C$ , the total amount of computation available. While the device is running the tasks, the *tradeoff mechanism* maintains state variables  $b_m(l)$ , which are estimates of the *benefit* offered by each particular implementation level of each task. This benefit represents an abstract

quantity whose meaning is dependent on our overall objective in allocating computational resources.

In a given time period, implementation levels  $l_m$  have been chosen. After all observations are made for the time period, we assess the quality of each task through a function  $Q_m$ . These observations are then used to adjust the implementation levels of the tasks. We use a two phase method to do this adjustment. First, based upon the quality assessments  $Q_m$ , we adjust benefit estimates  $b_m(l)$  for the implementation levels of each task. Then, we solve a *multiple-choice knapsack problem* to determine which implementation levels  $l_m$  are chosen for the next time period. More specifically, we try to choose  $l_m$  for all tasks in order to solve the following optimization problem:

$$\text{maximize} \quad \sum_{i=1}^M b_i(l_i) \quad (1)$$

$$\text{subject to} \quad \sum_{i=1}^M c_i(l_i) \leq C \quad (2)$$

### 3. APPLICATION TO MOTION ESTIMATION

We apply our system to motion estimation for standards-based video compression by treating the motion estimation for *each* macroblock in a predicted frame as a separate task. The computational resource being allocated is the total number of search positions that can be used to find one match for each of the macroblocks. Implementation level  $l$  corresponds to a search area of  $[-l, l] \times [-l, l]$ , which costs  $(2l + 1)^2$  search positions. Currently, we only consider exhaustive searches where every location in a search area is considered.

We expect the matching error or minimum SAD of the estimator to be at a minimum when the estimator is at the highest level implementation, corresponding to the greatest search size area. The smaller the search size is, the greater the resulting matching error will be. Therefore, we define the quality of the estimators  $Q_m$  to be the negative of the SAD for a given frame. After each frame is encoded, we modify  $b_m(l)$  for all implementation levels  $l$  for each task  $m$  as such:

- Find  $c = \max(|x|, |y|)$  where  $x$  and  $y$  are the horizontal and vertical components of the estimated displacement or motion vector.
- Set  $b_m(c)$  to be  $Q_m$ .
- For all  $l < c$ : If  $b_m(l)$  is greater than  $Q_m - d$ , set  $b_m(l)$  to be  $Q_m + d$ .
- For all  $l > c$ : If  $b_m(l)$  is greater than  $Q_m + d$ , set  $b_m(l)$  to be  $Q_m - d$ .

## 4. PRELIMINARY RESULTS

We integrated our adaptive motion estimation with the UBC H.263 encoder [2]. We encoded sequences both with our integrated version of the encoder and the standard version of the encoder. All sequences are standard CIF or SIF sized test sequences except for *Basketball* which was digitized from a television broadcast and *MIT* which originated from the Advanced Television Research Program at MIT. The standard version of the encoder was run with a search area of 4 pixels in each direction, and the integrated version of the coder was run with an equivalent complexity constraint of  $(2 \times 4 + 1)^2 MB = 81MB$  total search positions available, where  $MB$  is the number of macroblocks in the picture.

Sequence	Frame Rate (fps)	BitRate (kbps)	Standard PSNR	Integrated PSNR
Flower Garden	15	1100	28.20	28.68
Basketball	15	1100	29.55	29.99
MIT	30	1200	29.79	30.42
Mobile	15	900	29.38	29.37
Football	30	1300	29.79	29.77

The first three sequences have a lot of translational motion, while the last two have varied motion all across the frames. In the case of the first three, the system is able to reallocate more search positions to the areas of the frame with more motion. In the case of the last two, since all areas of the picture have significant motion, there is little improvement to be had in reallocating search positions from one macroblock to another.

## 5. CONCLUSION AND FUTURE DIRECTIONS

We have applied a framework for running multiple real-time tasks to the problem of motion estimation in video compression, and have demonstrated its usefulness in being able to “point out” areas of active motion within a frame and allocate more computational resources to motion estimation in those areas. In the future, we plan to refine our models of computational costs to more accurately reflect execution times on common processors, and also to consider different search strategies besides exhaustive search.

## 6. REFERENCES

- [1] J. Yeh and J. Wawrzyniec, “Quality Based Compute-Resource Allocation in Real-Time Signal Processing,” in *Conference Record of IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 2003.
- [2] F. Kossentini, M. Gallant, G. Cote, and B. Erol, “TMN Version 3.0 encoder,” University of British Columbia Image Processing Laboratory, May 1997.