Global Optimization for Mappping Parallel Image Processing Tasks on Distributed Memory Machines

CHeolwhan Lee, Yuan-Fang Wang, and Tao Yang
Department of Computer Science
University of California
Santa Barbara, CA 93106

Introduction

A graphical parallel programming tool for image processing.

- Visual programming interface through Khoros.
- Optimization of data mapping and task execution.
- Automated generation of parallel code.

Problem definition

- **Given** a chain of tasks with nested loops.
- **Map** tasks to processors, decide the computation and data distribution for each task.
- **Performance goal:** To minimize the overall parallel time.

Considerations:

- Exploiting both data and loop parallelism.
- Image size and #processor may vary at run-time.
- Computation weights may be data-dependent.

Assumptions

- **Computation and data mapping for each task.**

  Use one of HPF data distribution methods:

  row, column, row-cyclic, column-cyclic, block, and block-cyclic partitions

- **Computation follows chain-dependence.**

Related Work

- Parallel algorithms for image processing (e.g. Sahni).
- Dynamic scheduling and task mapping for image processing. (e.g. Jamieson, Siegel, Prasanna).
- Library-based parallel systems for image processing. (e.g. Jamieson, Reeves).
- Exploring task and data parallelism. (Banerjee, Subhlok)

Scheduling Task Chains with Loops

Optimization for different cases:

- Handling simple chains.
- Merging tasks for reducing complexity.
- Handing chains with loops.
- Considering the run-time variation of problem size and #processor.

The Algorithm

- **Step 1: Graph partitioning.**

  Traverse the tree control structure to identify a set of maximal subgraph chains which contain only data-independent tasks.

  For each of those chains, we apply the optimization technique described from Step 2 to Step 5.

- **Step 2: Graph reduction.**

  Merge pixel operations to reduce the number of task nodes.

- **Step 3: Loop linearization.**

  Traverse the tree control structure of each chain in a bottom-up manner to linearize loops.

- **Step 4: Constrained shortest path searching.**

  Construct a scheduling graph for each parameter setting and derive the optimal assignment.

- **Step 5: Scheduling for optimal average performance.**

  Construct an augmented scheduling graph for each chain and derive the mapping which has optimal average performance.

Theoretical Properties

Theorem: Assume loop iteration numbers are large. For each maximal data-independent subgraph chain with loops, the above algorithm finds a mapping which is asymptotically optimal in terms of the average performance (MPR).

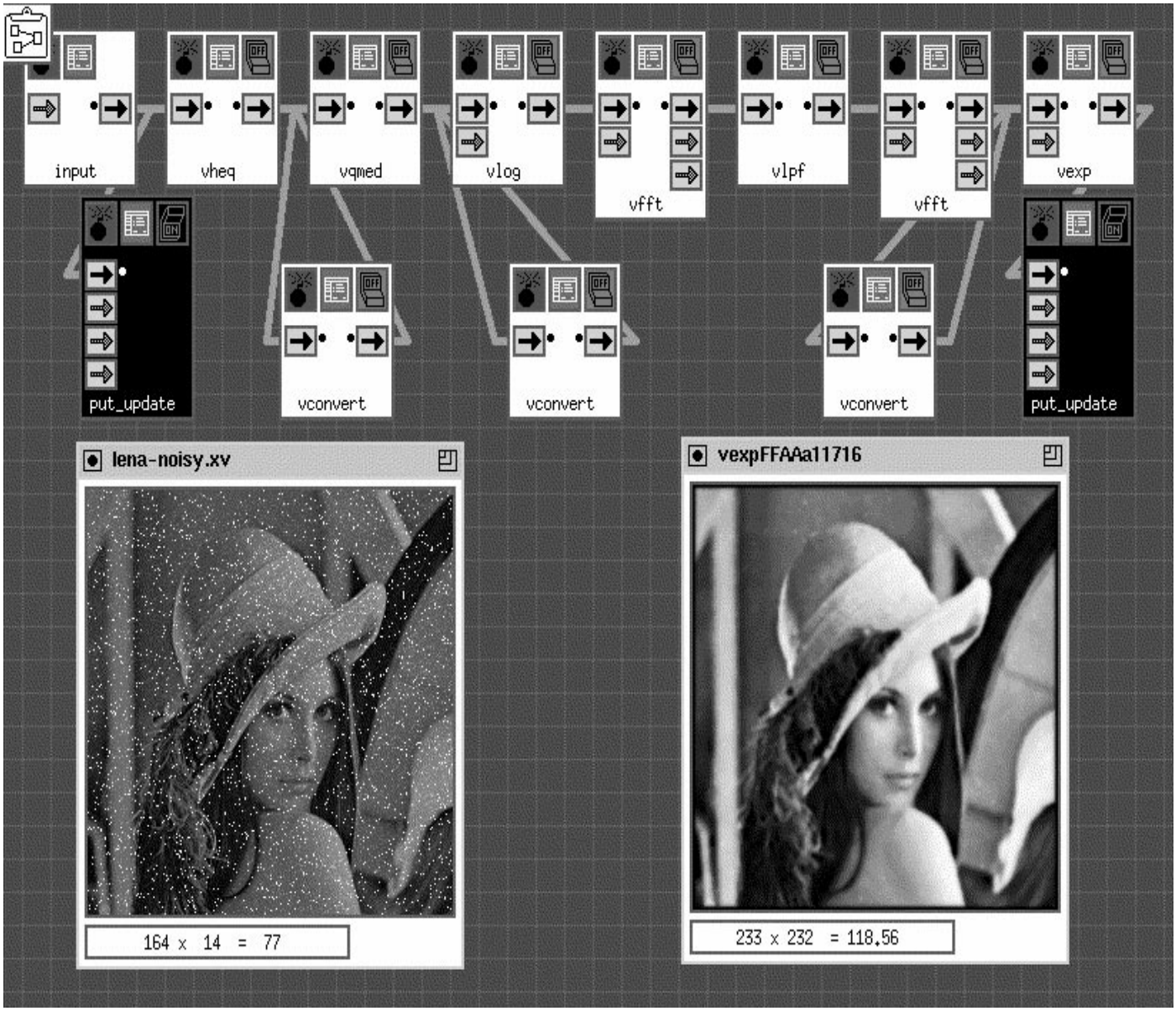


Figure 1: An image processing operation constructed using `Khoros`. The image shown at bottom left is the input image, and The image shown at bottom right is the output image from the operation.

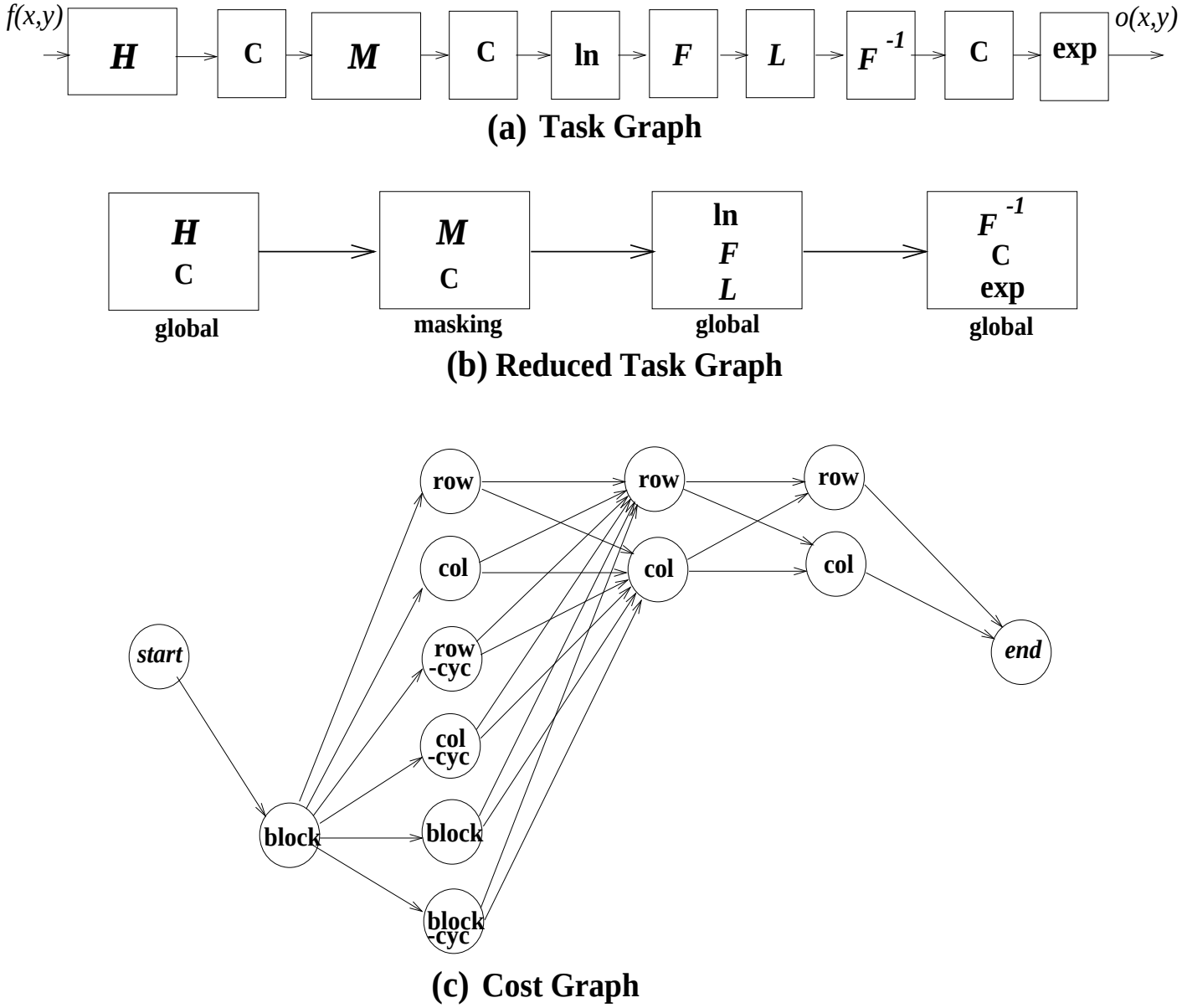
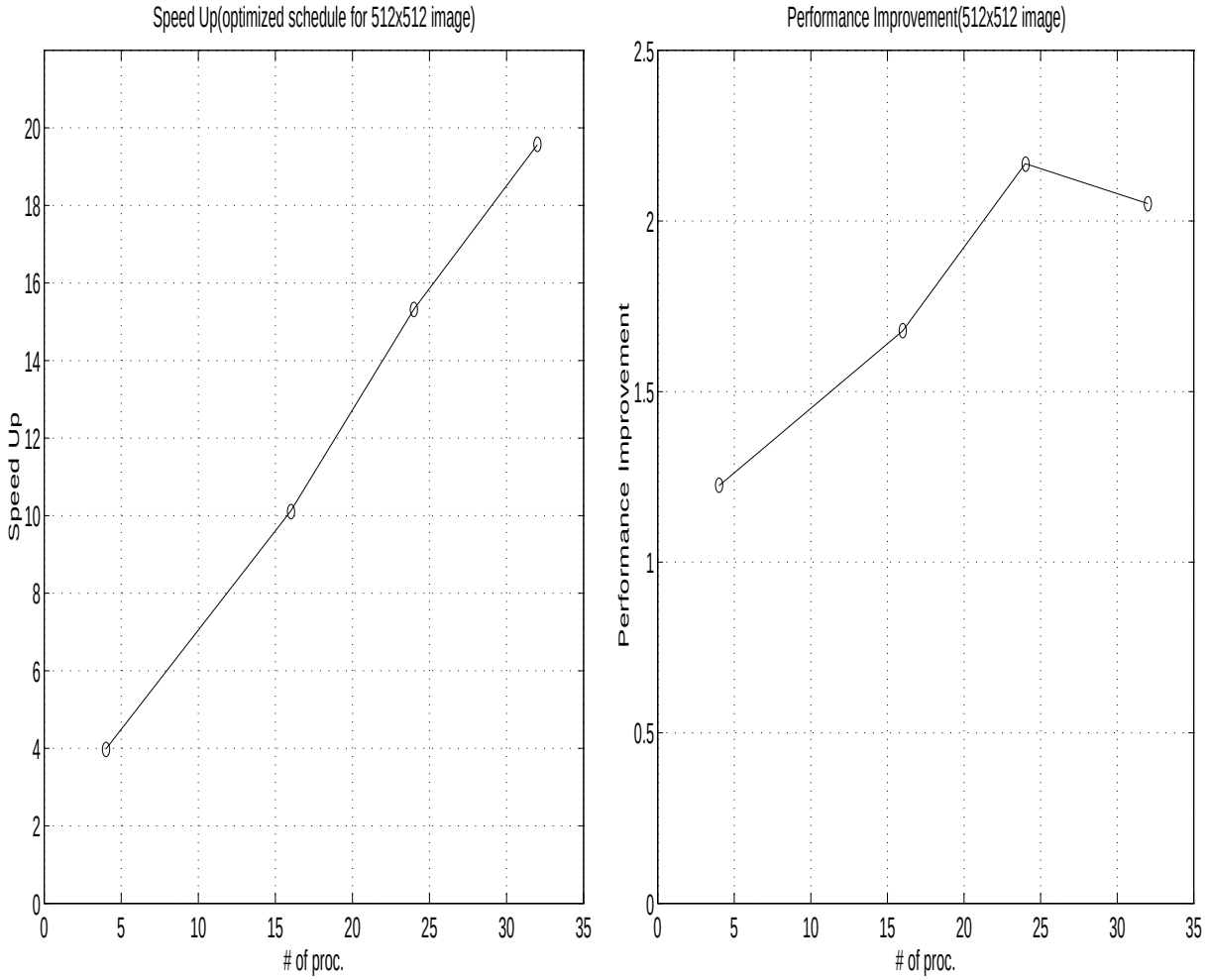


Figure 2: (a) The task graph for the noisy-reduction filter, (b) the reduced task graph, and (c) the cost graph.



(a) (b)

Figure 3: (a) The speedup of the optimized parallel program over the sequential program on MEIKO CS-2. (b) The performance improvement of the optimized codes over unoptimized codes on MEIKO CS-2.

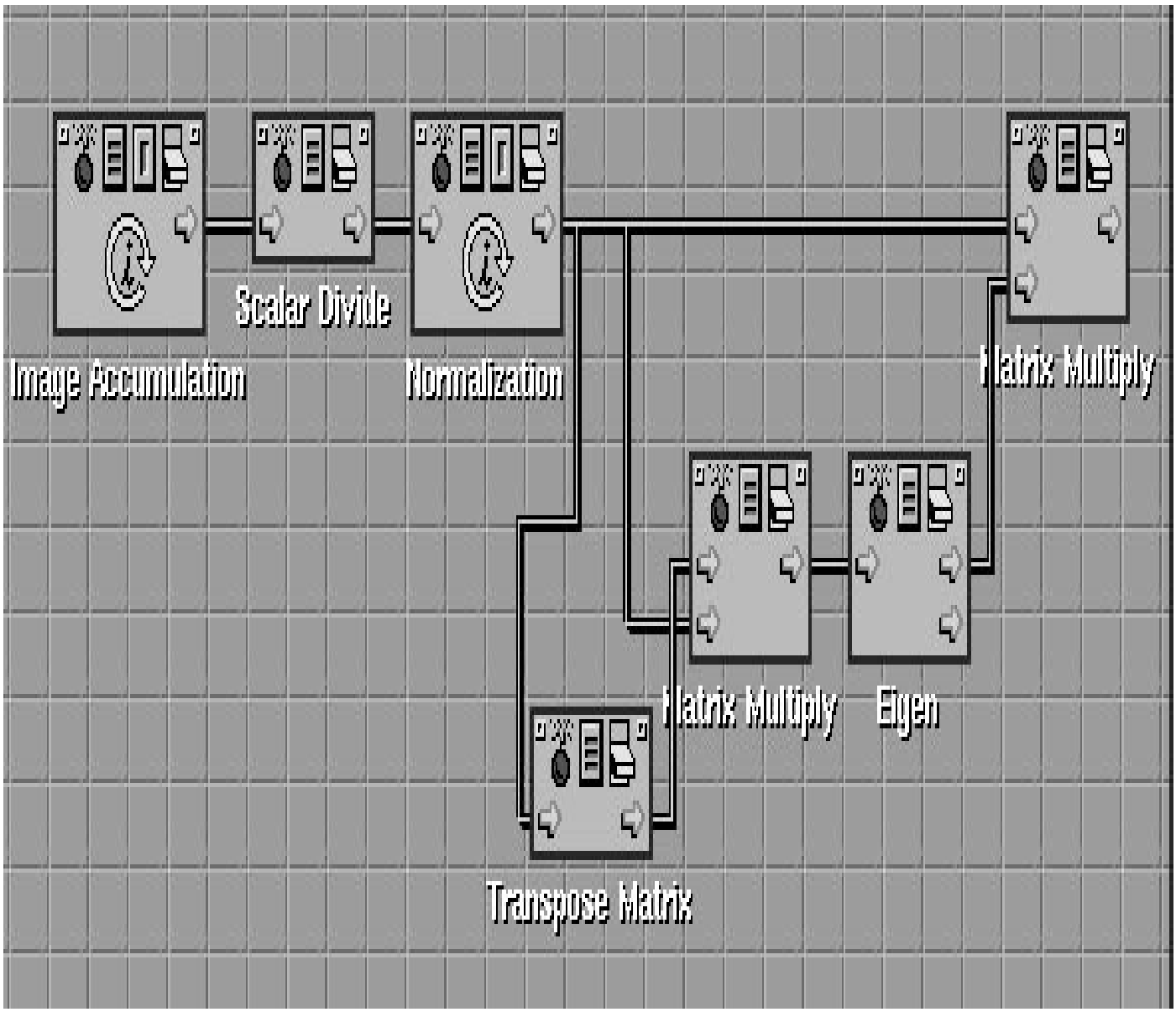


Figure 4: Implementation of the eigenface recognition algorithm using `Khoros`.

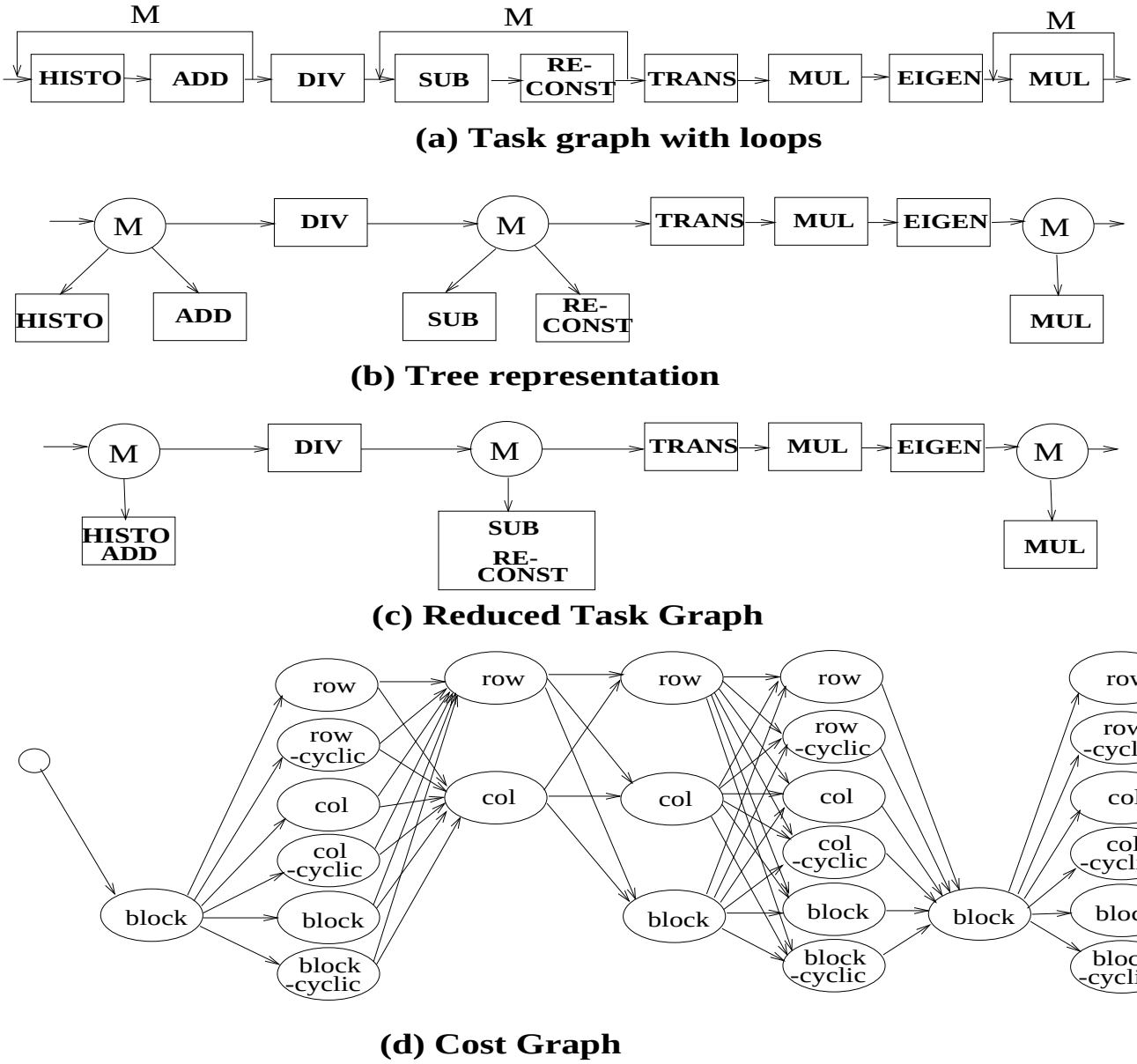
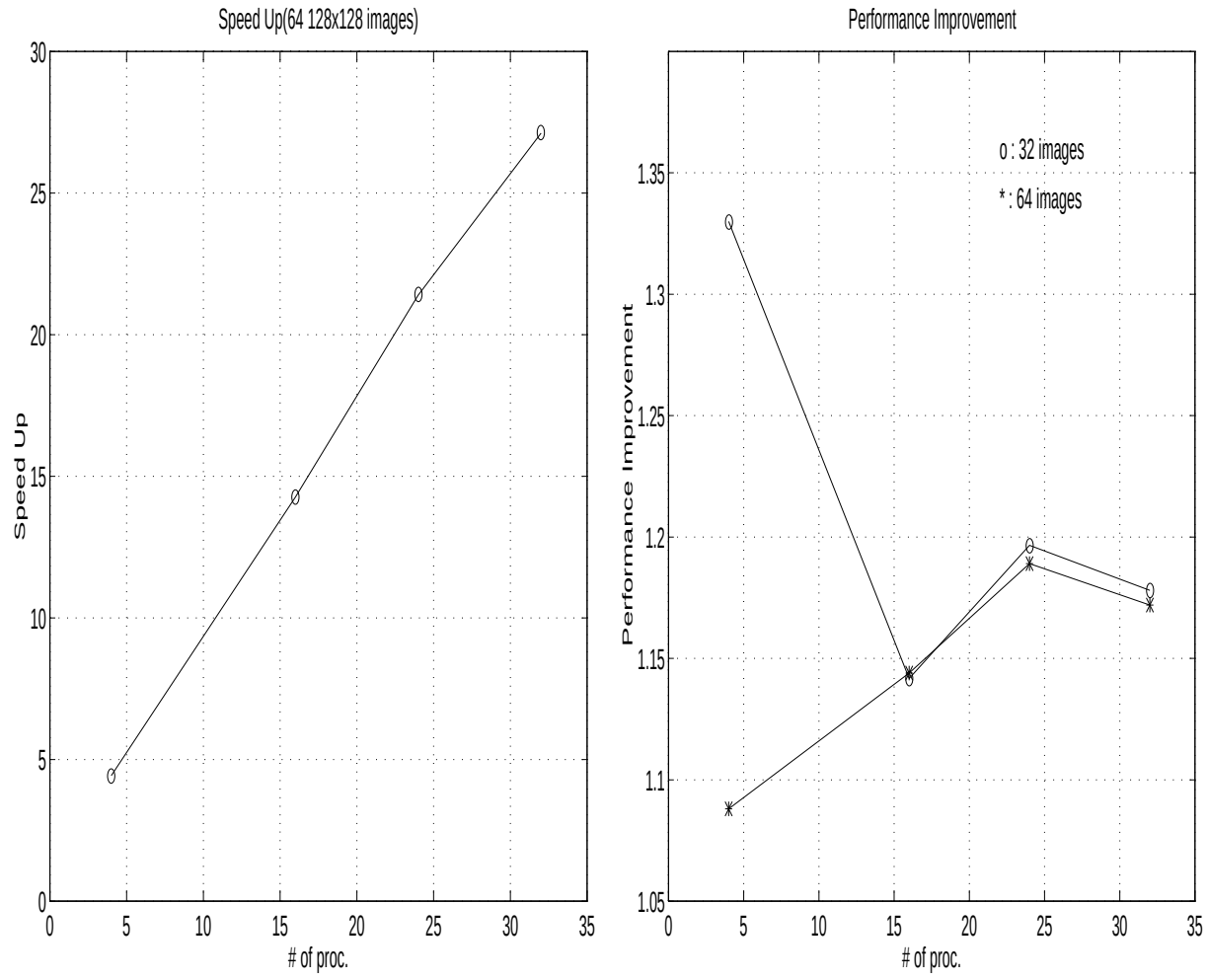


Figure 5: Graph representations of the eigen image computation.



(a) (b)

Figure 6: (a) The speedup of the parallel eigenface computation on MEIKO CS-2 using 64 128×128 face images. (b) The performance improvement of the parallel eigenface computation using 32 64×64 face images and 64 128×128 face images on MEIKO CS-2. $p = 4, 16, 24$ and 32.