Parallel Implementation of a Unified Approach to Image Focus and Defocus Analysis on the Parallel Virtual Machine

Yen-Fu Liu, Nai-Wei Lo, Murali Subbarao, Bradley S. Carlson
yfliu@sbee.sunysb.edu, naiwei@sbee.sunysb.edu
murali@sbee.sunysb.edu, bcarlson@sbee.sunysb.edu
State University of New York at Stony Brook
Stony Brook, New York 11794-2350, USA

ABSTRACT

A unified approach to image focus and defocus analysis (UFDA) was proposed recently for three-dimensional shape and focused image recovery of objects. One version of this approach which yields very accurate results is highly computationally intensive. In this paper we present a parallel implementation of this version of UFDA on the Parallel Virtual Machine (PVM). One of the most computationally intensive part of the UFDA approach is the estimation of image data that would be recorded by a camera for a given solution for 3D shape and focused image. This computational step has to be repeated once during each iteration of the optimization algorithm. Therefore this step has been sped up by using the Parallel Virtual Machine (PVM). PVM is a software package that allows a heterogeneous network of parallel and serial computers to appear as a single concurrent computational resource. In our experimental environment PVM is installed on two UNIX workstations communicating over Ethernet to exploit parallel processing capability. Experimental results showed that the communication overhead in this case was comparatively low. An average 1.93 speedup is gained by the parallel UFDA algorithm running on PVM platform compared to the execution time of sequential processing. This illustrates a practical application of PVM to 3D machine vision.

Keywords: unified focus and defocus analysis, optimization, parallel virtual machine, 3D shape measurement .

1 Introduction

The recovery of three-dimensional (3D) scene information of objects from images is a problem of inverse optics. Methods to solve this problem normally involves intensive computation in modeling the image formation process. Developing a highly accurate model for the image formation process that requires a reasonable amount of computation is an important problem. One way to handle this problem is to use a parallel algorithm implemented on a parallel computer. In this paper we present a parallel implementation of the Unified Focus and Defocus Analysis (UFDA)

[11] for 3D infomation recovery on a local area network (LAN). The communication interface between different machines is provided by the Parallel Virtual Machine (PVM) [3,14]. PVM is a portable message-passing programming system, designed to link separate host machines to form a "virtual machine" .

UFDA was proposed by us recently [11] for accurate reconstruction of 3D shape and focused image from a sequence of defcoused images. This technique unifed two distinct approaches – Image Focus Analysis (IFA)[4,5,10] and Image Defocus Analysis (IDA)[1,9]. The unification of IFA and IDA to UFDA is achieved by exploiting the relationship between the number of constraints embodied in the defocused images of these two approaches to the number of unknowns in the 3D shape and focused image. The framework of UFDA is based on modeling the sensing of defocused images in a camera system. A "Three-Dimensional Point Spread Function" (3D PSF) in the (x, y, d) space for this image formation model is introduced. Here x and y are the image spatial coordinates and d is a parameter representing the level of defocus. The problem of 3D shape and focused image reconstruction is formulated as an optimization problem where the difference (mean-square error) between the observed image data and the estimated image data is minimized. The estimated image data is obtained from the image sensing model and the current best known solutions to the 3D shape and focused image. An initial estimation to the solutions is obtained through traditional shape-from-focus methods. This solution is improved iteratively by a gradient descent approach. As we showed in [11], this is a very computationally intensive process, specially the part of estimating image data for a given solution to the 3D shape and focused image. This part has to be repeated to update the estimated image data during each iteration of the gradient descent search. There are at least two ways to handle this particular computational problem. One is to optimize the gradient descent approach in order to reduce the amount of computation. The other is to develop a corresponding parallel algorithm running on multiple machines/microprocessors to speed up the process. One efficient optimization technique to UFDA has been proposed in [12]. This paper presents the parallel version of UFDA implemented on PVM. Our experiments show that PVM provides a means to distribute heavy computation load of practical image recovery application on a LAN and improve the performance with respect to the total execution time.

2 3D shape and focused image recovery by UFDA

2.1 UFDA

UFDA [11] is a new theory that unified IFA and IDA. To briefly illustrate IFA and IDA in terms of the definition of the problem for the 3D shape and focused image recovery, consider the following. The image coordinates of a point light source is denoted by (x', y') and the image coordinates of a point where the brightness is measured on the image detector is denoted by (x, y). The camera parameters are $\mathbf{e} = (D, f, s)$ where D is the diameter of camera aperture, f is the focal length of the lens, and s is the distance between the lens and the image detector. An image sequence can be thought of as sampled data of an image volume and denote the image data as $g(x, y, d)$ for $x = 0, 1, 2, \cdots, J-1$, $y = 0, 1, 2, \cdots, K-1$, $d = 0, 1, 2, \cdots, I-1$, where J and K are the number of rows and columns respectively in each image frame and I is the number of image frames. Whereas IFA uses many image frames, IDA uses only $2 \sim 3$ images from this image sequence. $d_v(x', y')$ represents the 3D shape and $F(x', y')$ is the focused image. The problem for IFA can be stated as– given the image sequence $g(x, y, d)$ and the camera parameters s and f, find $d_v(x', y')$ and $F(x', y')$ in some image region or over the entire image. Similarly, the problem for IDA can be stated as– given two of the images $g_m(x, y)$ and $g_n(x, y)$ in the image sequence $g(x, y, d)$, the camera parameters $\mathbf{e}_m$ and $\mathbf{e}_n$ corresponding to these two images, and the the camera's point spread function as a function of the camera parameters, find $d_v(x', y')$ and $F(x', y')$ in some image region or over the entire image.

From the above statements, the IFA and IDA methods can be clearly viewd as two extremes of a range of methods useful in 3D shape and focused image recovery. At one end of this range of methods is the IFA method which uses a large amount of image data but minimal information about the camera characteristics (e.g. the camera's point spread function). At the other end is the IDA method which uses minimal image data but much information about the camera characteristics. Hence the theory of UFDA results in a unified approach that suggests new methods that lie between the two extremes of the IFA and IDA methods. To this point, the unknowns in both the IFA and IDA methods are $d_v(x', y')$ and $F(x', y')$. If the image frame size is J rows and K columns, then the number of unknowns for $d_v(x', y')$ and $F(x', y')$ are both $J \times K$. Therefore the total number of unknowns is $2JK$. For UFDA to determine these unknowns, a number of image frames $g(x, y, d)$ are recorded at different camera parameter settings. This recorded image volume data depends on the camera parameters $\mathbf{e}$, the 3D shape $d_v(x', y')$, and the focused image $F(x', y')$. This dependence is specified by the camera characteristics (e.g. point spread function) [8]. Hence, a three-dimensional point spread function (3D PSF) $h(x, y, \mathbf{e}, d)$ [11] that is based on the paraxial geometric optics is developed in the UFDA. Now the problem of 3D shape and focused image recovery for UFDA can be stated as – given a sequence of images obtained by sampling $g(x, y.d)$ and the proposed image formation model, find $d_v(x', y')$ and $F(x', y')$.

Under certain weak conditions we can derive [11] a three-dimensional convolution expression:

$$g(x, y, d) \;=\; \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F'(x', y', d')\; h(x - x', y - y', d - d')\; dx'\; dy'\; dd'. \tag{1}$$

where

$$F'(x', y', d') = \begin{cases} F'(x', y') & \text{if } d' = d_v(x', y') \text{ and} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The above convolution expression can be abbreviated as

$$g(x, y, d) \;=\; F'(x', y', d') \;\star\; h(x', y', d') \tag{3}$$

where $\star$ denotes the convolution operator.
Details on the 3D PSF and the derivation of 3D convolution equation are reported in [11].

The estimated image data $g_e(x, y, d)$ is obtained from Eq. (1) using the 3D PSF h and the current best known solutions to the 3D shape $d_v(x', y')$ and focused image $F(x', y')$. Then the problem of 3D shape and focused image reconstruction is formulated as an optimization problem where the difference or mean-square error E between the observed image data $g_o(x, y, d)$ and the estimated image data $g_e(x, y, d)$ is minimized

$$E = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (g_o(x, y, d) - g_e(x, y, d))^2 \, dx \, dy \, dd \tag{4}$$

An initial estimation of the solutions is obtained through traditional IDA and/or IFA methods. This solution is improved iteratively by an optimization technique.

2.2 Optimization

In the optimization, a gradient descent approach is used to minimize the sum of squared error between the observed image data and estimated image data and update the estimated solution iteratively. This method is based on the iterative gradient descent approach of going downhill with respect to the error function to find the lowest point. The focused image surface in a small image region is approximated by a piecewise planar surface patch with three parameters– slope with respect to x-axis, slope with respect to y-axis, and z-axis intercept. Error gradient with respect to these three parameters were used in the gradient descent error minimization. A sequential followed by a parallel parameter search (SPPS) was used for this gradient descent method. First, the optimization was done with respect to one parameter at a time. After that, three parameters were searched simultaneously.

2.3 Sequential Implementation

Three image models ($16 \times 16 \times 16$, $32 \times 32 \times 32$ and $64 \times 64 \times 32$ size image volume data) were synthesized where the focused image surface of each model was a hemispherical object (with radius 12,24,24) and the focused image was a checker board. From UFDA, the observed image data $g(x, y, d)$ was synthesized using Eq.(1) with camera parameters (D=9mm, f=35mm, s= 35mm to 36.5mm). Then an initial solution for the focused image and the estimated 3D shape of these image models were obtained using an IFA method. In the IFA method a focus measure (energy of image Laplacian) was computed in a small window of size 4×4 (for $16 \times 16 \times 16$) and 8×8 (for $32 \times 32 \times 32$ and $64 \times 64 \times 32$) non-overlapping regions. A piecewise constant approximation to the focused image surface in each of the above window region was obtained by finding the position where the focus measure was a maximum. These estimated solutions and the 3D PSF were used to compute the estimated image data $g_e(x, y, d)$ using Eq.(1). The error E between the observed image data $g_o(x, y, d)$ and the estimated image data $g_e(x, y, d)$ was computed using Eq.(4). The SPPS gradient descent method is applied to improve the estimated solution iteratively.

This UFDA approach reduces the errors in shape and focused image introduced by the image-overlap problem and the non-smoothness of the object's 3D shape. Experimental results were reported in [11] where accurate reconstructed 3D shape and focused image were attained at the cost of computation.

3 Parallelizable Portion of UFDA

The highly intensive computation load of UFDA approach is mainly due to the estimation of image data in the image formation process iteratively using Eq.(1). The algorithm of convolving estimated 3D object with 3D PSF is described in the following.

```
for i = 0 to N do                /* i: frame index */
    for j = 0 to N do            /* j: row index */
        for k = 0 to N do        / * k: column index */
            sum= 0;
            for m = 0 to N do        /* m: row index */
                for n = 0 to N do        /* n: column index */
                    sum = sum + F(m,n)· h(i − d_v(m,n), j − m, k − n);
                end/* n loop */
            end/* m loop */
            if (sum>255)
                then sum = 255;
            g(i,j,k) =sum;
        end                      /* k loop */
    end                  /* j loop */
end              /* i loop */
```

where the $g(i,j,k)$ (i.e. sum) denotes the gray level at a particular image coordinate (i,j,k) in the image volume, $h(i,j,k)$ is the 3D PSF corresponding to (i,j,k), $d_v(m,n)$ is the focused image surface at (m,n), $F(m,n)$ is the gray level on the $d_v(m,n)$, N represents the total number of image frames and image size for the corresponding index repectively. This routine explains explicitly the way we do convoluton for a shift variant type point spread funcion. From the accuracy point of view this image sensing model takes into account the contribution from all object point sources in estimating the observed image brightness at a given point. In particular, the image brightness near the border of an image region is computed by taking into account all possible object point sources– both those inside the image region and those outside but close to the border. Therefore the error in 3D shape recovery is minimized. However, the price paid to minimize the error is heavy computation workload. In this UFDA algorithm for each image point in the image volume of size $I \times J \times K$ it has to compute all the possible blur values that are contributed by object surface of size $m \times n$. To complete this process a complexity order of $O(n^5)$ computation for each updated solution at each point is necessary. Since the estimated solution will update in each iteration during the optimization process, these loops will be executed repeatly.

After examining the structure of UFDA algorithm, data parallelization of the 3D object estimation portion of UFDA algorithm is obviously the suitable direction to develop a parallel version of UFDA algorithm from the sequential one.

4 Parallel Implementation of UFDA

In this section we describe the structure of Parallel Virtual Machine, PVM environment in our laboratory, and implementation of the parallel UFDA algorithm.

Parallel Virtual Machine (PVM) [13], started in the summer of 1989 at Oak Ridge National Laboratory (ORNL), is a software system that permits a network of heterogeneous UNIX computers to be used as a single large memory-distributed parallel computer. Therefore, the computation power of a cluster of workstations/computers can be aggregated to solve large computational problems. In [2] for many large scientific applications the viability of network computing on a cluster of workstations has been established. In this paper we are interested in the viability of network computing for practical applications such as UFDA.

PVM provides the functions to start up tasks on the virtual machine and allows the tasks to communicate and synchronize with each other. A computation unit in PVM is defined as a *task* usually analogous to a UNIX process. Applications written in Fortran77 or C can be parallelized by using message-passing constructs. Thus under careful design an application can cooperate multiple tasks in parallel to solve a problem by sending and receiving messages between tasks (processes).

In order to function correctly in a heterogeneous computer environment PVM handles all data conversion between computers with different data formats if necessary. PVM also allows application tasks to exploit the architecture best suited to their solution and provides heterogeneity at the application, machine, and network level.

The PVM system contains two parts – a daemon and an interface library. Any user with a valid login can install the daemon on a machine. When a user wants to run a PVM application, he first creates a virtual machine by starting up PVM. The PVM application can then be executed from a UNIX prompt on any of the hosts registered in the virtual machine environment. Multiusers and multitasking are supported in the PVM system. The PVM library includes user callable routines for message passing, spawning processes, coordinating tasks, and modifying the virtual machine. In order to use PVM system all application programs must be linked with this library.

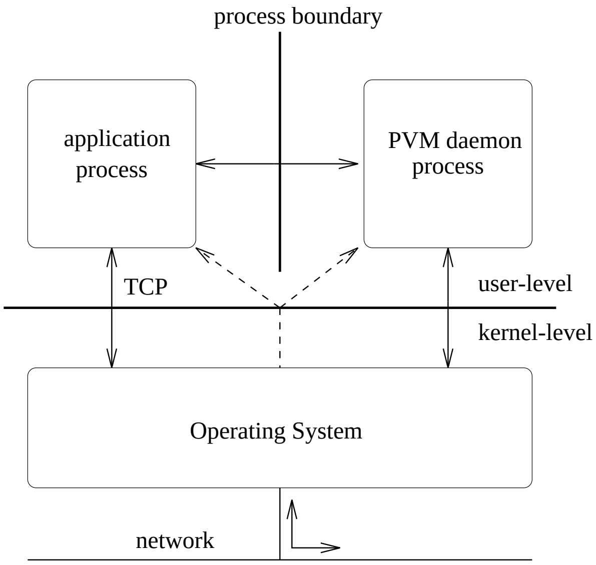

Figure 1: The structure of PVM system. Data paths are indicated by solid arrow lines and scheduling control paths aremarked by broken arrow lines.

The PVM system adopts two-process setting technique. Fig. 1 shows the details. The PVM daemon process is responsible for only communication. Scheduling of the application process and communication process is done by the operating system's scheduler. To exchange messages between two application processes the data is copied across the process boundary into and out of PVM daemon process, and communicating the messages across the network by means of the **UDP/IP** protocol. The PVM daemon process is responsible for retransmitting lost **UDP** packets and resolving reordering of packets to provide reliable communication. In the recent versions of PVM system a direct **TCP** connection between two communicating application processes is established for message passing. This implementation reduces the total communication latency by a factor of 3-4. In [6,7] new designs using multithread implementation to further hide the communication latency are discussed.

The PVM environment in our experiment is composed of two SUN SPARC IPX workstations with corresponding 16 and 32 megabyte main (RAM) memory on each machine connected by 10 megabits/sec Ethernet. Each computer has its own hard disk drive. The parallel programs include a master part and a slave part. The master program is implemented on the SUN SPARC IPX with 32 megabyte memory on board and the slave program is installed on the other computer. Once PVM daemon processes on both machines are activated by executing the PVM console program from the master machine, the master parallel program is executed and its corresponding slave parallel program is invoked through the network automatically. All data communication work is handled by PVM connection function calls. For example, **pvm_send()** and **pvm_recv()** are the functions to send and receive data between two application processes respectively.

To utilize the computation power of local area network efficiently data communication over-

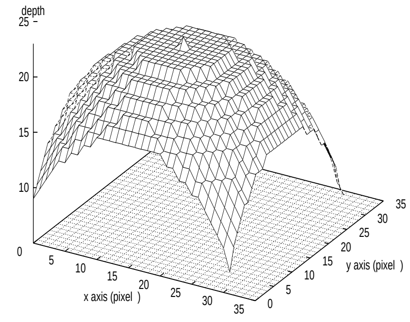

Figure 2: The original $32 \times 32 \times 32$ focused image surface (FIS).

head among computers must be reduced to a minimum extent. In our implementation of the parallel estimation algorithm initial data is transmitted over the network from the master machine to the slave machine at the first iteration of estimation. Initial data includes estimated focus image, observed image volume, estimated 3D shape and precomputed 3D PSF data. For the regular data communication only the current estimated 3D shape and new estimated image volume are transmitted back and forth at each iteration of estimation.

To get significant performance improvement from parallel execution the computation complexity of parallelized algorithm (process) should be at least two orders of magnitude greater than the complexity of communication overhead. From section 3 we know the computation complexity of the sequential estimation process is $O(n^5)$ where n is the number of image frames. Also the complexity of communication overhead, $O(n^3)$, is bounded by the transmission process of new estimated image volume at each iteration. Therefore, we predict that the overall execution speed of the 3D machine vision application can be improved by performing parallel implementation.

5 Experimental Results

The first important issue for a parallel application/experiment is to generate correct results as its sequential counterpart does. In our UFDA experiments the output results from both parallel and sequential versions were both the same. In the $32 \times 32 \times 32$ image volume case the original focused image surface (FIS), the initial solution obtained from IFA method and the recovered focused image surface are shown in Figs. 2, 3 and 4, respectively.

In order to eliminate the performance inconsistency and degradation of both parallel and

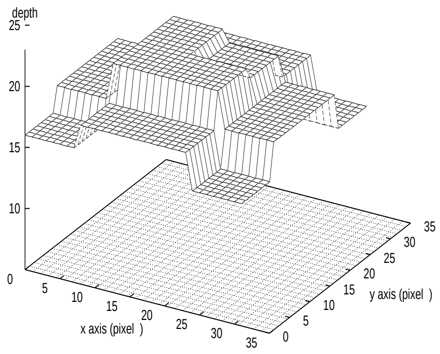

Figure 3: The initial solution of $32 \times 32 \times 32$ focused image surface generated by IFA method.

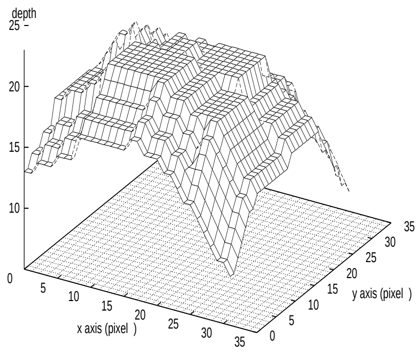

Figure 4: The recovered $32 \times 32 \times 32$ focused image surface generated by parallelized UFDA. (same as sequential version)

sequential UFDA applications caused by the dynamic network workload all performance measurements are recorded in which only the UFDA user process(es) and essential system daemon processes run on the workstation(s) of experiment network.

In Table 1 the performance details of sequential UFDA under different sizes of image volume are presented. Notice that the total execution time is equal to the user time plus the system time. For the parallel UFDA Table 2 shows the corresponding performance measurements. In parallel UFDA the total execution time is composed of user time, system time and communication overhead. The user time is defined as the execution time spent by the user process. The definition of system time is the execution time spent by the system (UNIX kernel). The communication overhead is caused by the internal data processing delay of ethernet cards and the transmission latency of the local network. All measured entries in Table 1 and Table 2 are reported in seconds. The results in Table 2 are measured from the workstation as the master machine in PVM.

Table 1	Sequential Implementation		
Image Volume	Total Execution Time	User Time	System Time
$16 \times 16 \times 16$	70.55	70.35	0.2
$32 \times 32 \times 32$	2345.15	2344.4	0.75
$64 \times 64 \times 32$	35421.5	35406	15.5

Table 2	Parallel Implementation			
Image Volume	Total Execution Time	User Time	System Time	Communication Overhead
$16 \times 16 \times 16$	56.35	38.3	2.1	15.95
$32 \times 32 \times 32$	1242.75	1168.8	2.85	71.1
$64 \times 64 \times 32$	18009.2	17784.8	21.1	203.3

Compared to Table 1 the values of user time in Table 2 are decreased because the computation workload is shared by another workstation. On the contrary the values of system time in Table 2 are greater than the ones in Table 1. The reason is PVM procedure calls invoke some kernel functions. Therefore, the system time increases when more PVM procedure calls are executed.

To describe the speed advantage of a parallel algorithm compared to a serial reference algorithm the speedup ratio must be defined. Let m represent the problem size (that is, the size of image volume in our case.). Suppose that we have a parallel algorithm that uses p processors and that terminates in time $T_p(m)$. Let $T(m)$ be the time required by the serial (uniprocessor) reference algorithm for this problem. The speedup of the parallel algorithm is defined as $S_p(m) = \frac{T(m)}{T_p(m)}$. The speedup ratios of parallel UFDA algorithm varied from 1.25 to 1.97 as shown in Fig. 5. The speedup diagram indicates that if the image volume is large enough, the communication overhead will not become the dominant factor of the total execution time. The percentage of total execution time spent on communication overhead under different image models presented in Fig. 6 also reveals the same point of view. The communication overhead is significant in $16 \times 16 \times 16$ image model case of parallel UFDA algorithm because the total computation workload is not large enough to hide the communication latency caused by sending initial

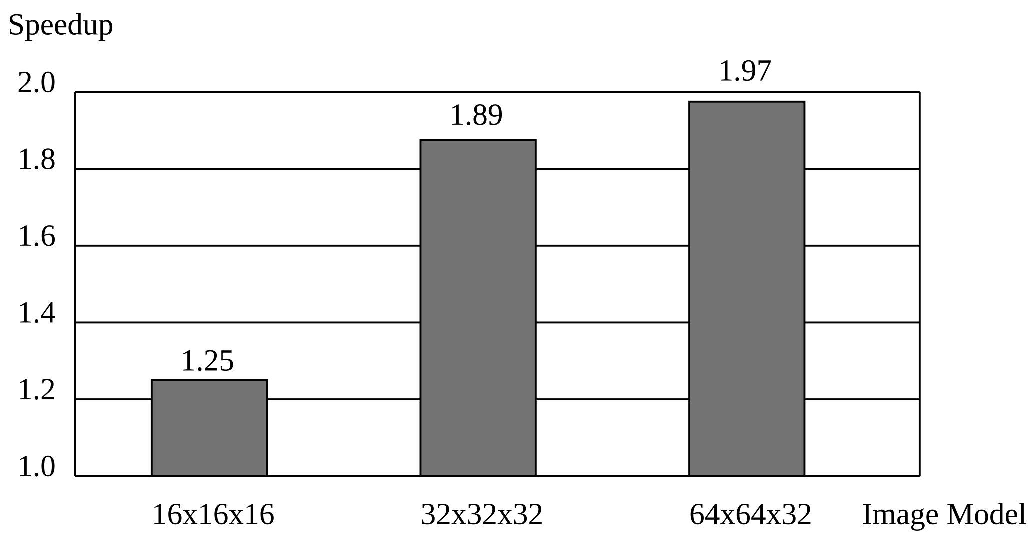

Figure 5: Speedup analysis under different image models.

data from the master machine to the slave machine. However, since the computation complexity of parallel UFDA algorithm $O(n^5)$ is two orders of magnitude greater than its corresponding communication complexity $O(n^3)$, once the image volume is large enough the communication overhead of inital data transmission between two workstations can be shared and hidden by each computation iteration.

From our experiments, we show the potential of utilizing the computation power of local area network to improve the performance of computation-intensive applications such as UFDA algorithm. When more processors are introduced in PVM system, the master/slave architecture will produce proportional communication overhead into the network. Therefore, implementation schemes and the architecture of parallel algorithms become more critical to further reduce total overhead and reach high speedup.

6 Conclusion

In this paper a parallel implementation of the UFDA using PVM is presented. One important characteristic of UFDA is the trade-off between the accuracy of 3D shape and focused image recovery and the data computation load. Higher accuracy is obtained at the cost of additional image acquisition and processing. PVM helps to speed up the most computationally intesive part of UFDA– the computation of estimated image data for a given solution.

Parallization of UFDA through PVM offers a possible solution to reduce the total execution time of the UFDA application and provides an example of utilizing the distributed computation power of local area network. In our experiments with two workstations networked, the parallel

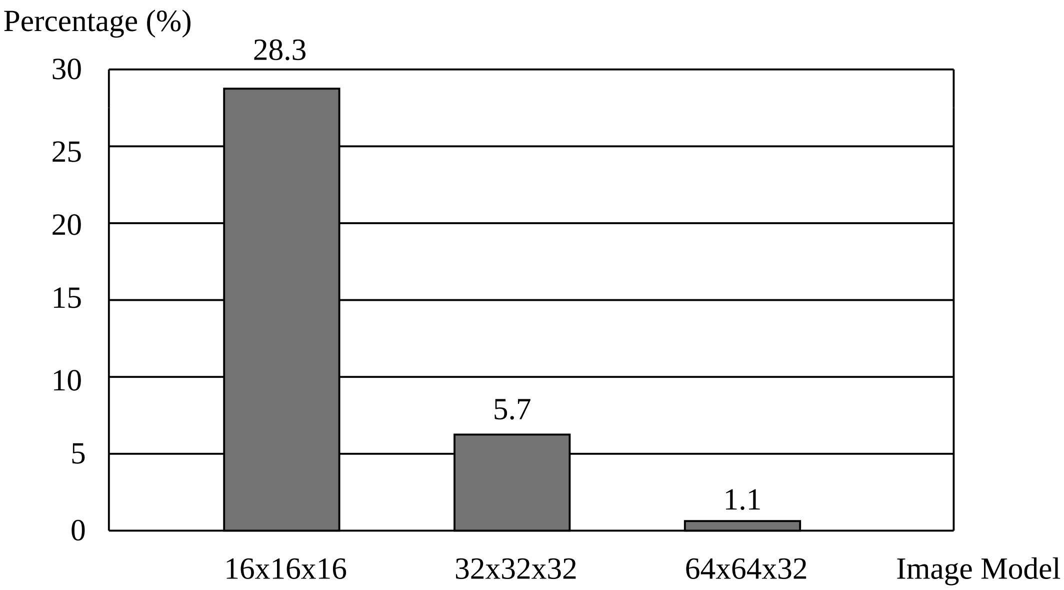

Figure 6: The percentage of total execution time spent on communication overhead in different image models.

UFDA application reaches a speedup in the range of 1.25 to 1.97 depending on size of image data. Our experiments demonstrate the advantage of using PVM to solve computation intensive problems in machine vision applications such as UFDA.

7 References

[1] J. Enns and P. Lawrence, "A Matrix Based Method for Determining Depth from Focus", *Proceedings of the IEEE Computer Society Conference on CVPR*, June 1991.

[2] G.A. Geist, B.W. Peyton, W.A. Shelton, and G.M. Stocks, "Modeling High-temperature Superconductors and Metallic Alloys on the Intel iPSC/860", *Proc. Fifth Distributed Memory Computing Conference, ed D. Walker and Q. Stout, IEEE Computer Society Press*, pp 504-512, April 1990.

[3] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine - A Users Guide and Tutorial for Network Parallel Computing,* MIT Press, 1994.

[4] E. Krotkov, "Focusing", *International Journal of Computer Vision*, 1, 223-237, 1987.

[5] S. K. Nayar and Y. Nakagawa, "Shape from Focus: an effective approach for rough surfaces", *IEEE Trans. on PAMI*, 16 (8):824-831, Aug. 1994.

[6] V. Strumpen and T. L. Casavant, "Implementing Communication Latency Hiding in High-Latency Computer Networks", *High-Performance Computing and Networking*, LNCS 919, pages 86-93. Springer-Verlag, Milano, Italy, May 1995.

[7] V. Strumpen, "Software-Based Communication Latency Hiding for Commodity Workstation Networks", *IEEE International Conference on Parallel Processing*, pages I-146 - I-153, 1996.

[8] M. Subbarao, and M. C. Lu, "Computer Modeling and Simulation of Camera Defocus", *Machine Vision and Applications*, (1994) 7, pp. 277-289.

[9] M. Subbarao and G. Surya, "Depth from Defocus: A Spatial Domain Approach", *International Journal of Computer Vision*, 13, 3, pp. 271-294 (1994).

[10] M. Subbarao and T. S. Choi, "Accurate Recovery of Three-Dimensional Shape from Image Focus", *IEEE Transactions on PAMI,* March 1995, pp. 266-274.

[11] M. Subbarao and Y.F. Liu, " Accurate Reconstruction of Three-dimensional shape and Focused Image from a Sequence of Noisy Defocused Images", SPIE Vol.2909 pp 178-191, Boston Mass. Nov. 1996

[12] M. Subbarao and Y.F. Liu, " Analysis of Defocused Image Data for 3D Shape Recovery using a Regularization Technique", SPIE Vol.3204 ,ISAM'97, Pittsburgh Oct. 1997

[13] V. Sunderam, "PVM: A Framework for Parallel Distributed Computing", *Concurrency: Practice and Experience*, Vol2 No. 4, December 1990.

[14] V. Sunderam, A. Geist, J. Dongarra, and R. Manchek, "The PVM concurrent computing system: evolution, experience, and trends", *Parallel Computing*, Vol. 20 (4), 1993.