Color-Based Object Recognition on a Grid

F.J. Seinstra and J.M. Geusebroek

ISLA, Informatics Institute, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{fjseins, mark}@science.uva.nl

Abstract. Multimedia data is rapidly gaining importance along with recent developments such as the increasing deployment of surveillance cameras in public locations, and the need for automatic comparison of forensic video evidence. In a few years time, analyzing the content of multimedia data will be a problem of phenomenal proportions, as digital video may produce data at rates beyond 100 Mb/s, and multimedia archives steadily run into Petabytes of storage space. Consequently, for urgent problems in multimedia content analysis, Grid computing is rapidly becoming indispensable.
This paper explores the viability of wide-area Grid systems in adhering to the heavy demands of a real-time task in multimedia content analysis. Specifically, we show the application of a robot dog, capable of recognizing objects from a set of 1,000 learned objects, while connected to a large-scale Grid system comprising of cluster systems in Europe and Australia. Our results indicate that we have reached the moment at which real-time image and video analysis on large-scale Grids is becoming a reality. Moreover, our approach shows the effective integration of state-of-the-art results from two largely distinct research fields: multimedia content analysis and Grid computing.

1 Introduction

This is the time in which information — be it scientific, industrial, or otherwise — is generally composed of *multimedia items*, i.e. a combination of pictorial, linguistic, and auditory data. The computerized access to the content of such information is recognized as a tremendous challenge [14], [23], [24]. For one, this is because the automatic deduction of semantics from multimedia data requires sophisticated techniques for data structuring, transformation, analysis, classification, and learning. From the nature of information creation, most of the data is irrelevant to a specific question. The challenge is therefore to discover and interpret tiny fractions of useful information in a whirlwind of meaningless noise. Due to the increasing storage and connectivity of multimedia data, automatic *multimedia content analysis* is becoming an ever more important area of research.

Multimedia content analysis (MMCA) considers all aspects of the automated extraction of knowledge from large multimedia data sets. From these, image and video data sets are by far the largest, and thus constitute the biggest challenge. Fundamental research questions in this area include:

– Can we automatically find (sub-)genres in images from a statistical evaluation of large image sets?
– Can we automatically learn to find objects in images and video streams from partially annotated image sets?

Scientifically, these questions deal with the philosophical foundations of cognition and giving names to things. From a societal perspective solutions are urgently needed, given the rapidly increasing volume of multimedia data.

Applications abound in which MMCA is essential. The Dutch Forensic Institute (NFI), for example, has an urgent need for detection of objects and individuals in video streams from surveillance cameras. The Netherlands Institute for Sound and Vision (Beeld&Geluid) aims to store digitized television broadcasts in large repositories, generating heavy demands on accessibility. The identification of materials and land-cover classes in Terabytes of hyper-spectral data obtained from NASA satellites is just one example from the scientific literature [30]. To adhere to the performance requirements of such colossal MMCA problems, parallel and distributed computing is rapidly becoming indispensable.

Even though many multimedia applications are ideal for parallel implementation [9], [10], [13], [26], [28], most multimedia researchers do not benefit from the many cluster systems available today. This is due to the lack of programming tools that can help non-expert parallel programmers in developing efficient high-performance multimedia applications. Existing tools generally require detailed parallelization knowledge beyond the expertise of the average user [15], [18], [22]. Hence, the key to effective support for high performance multimedia computing lies in the availability of a familiar (i.e. user transparent) *programming model* that hides the difficulties of parallel implementation from its users.

Having a familiar programming model available is only one part of a satisfactory solution. This is because in many emerging multimedia applications the use of distributed (Grid) resources is essential [20]. The combined use of such resources is hard, however, because these still lack the basic functionality needed for extensive use [17]. Obviously, to further stimulate Grid computing in the multimedia community, the abovementioned programming model must be supported by an easy-to-use *execution model*.

This paper describes our results obtained by matching an existing user transparent programming model (i.e. Parallel-Horus [18], [22]) with an execution model based on *Wide-Area Multimedia Services*, i.e. high performance multimedia functionality that can be invoked from sequential applications running on a desktop machine. We evaluate our approach by applying it to a state-of-the-art visual recognition task. Specifically, we present the application of a Sony Aibo robot dog, capable of recognizing objects from a set of 1,000 objects, while connected to a large-scale Grid system comprising of cluster systems in Europe and Australia. Our results indicate that we have now reached the moment at which real-time image and video analysis on large-scale Grids is becoming a reality.

This paper is organized as follows. Section 2 briefly introduces the Parallel-Horus architecture. In Section 3 we describe our services-based approach to wide-area multimedia computing. Section 4 discusses our robot dog application. Future work is discussed in Section 5. Concluding remarks are given in Section 6.

2 Parallel-Horus: An Overview

Parallel-Horus [19], [22], [18] is a cluster programming framework that allows programmers to implement parallel multimedia applications as fully sequential programs. The Parallel-Horus framework consists of commonly used multimedia data types and associated operations, implemented in C++ and MPI. The library's API is made identical to that of an existing sequential library: Horus [12]. The parallel functionality is integrated with Horus such that all existing sequential code remains unchanged. This approach has the major advantage that the important properties of the Horus library (i.e., maintainability, extensibility, and portability) to a large extent transfer to Parallel-Horus as well.

Similar to other frameworks [4], [13], Horus is based on the abstractions of Image Algebra [16], a mathematical notation for specifying image and video processing algorithms. Image Algebra recognizes that a small set of *algorithmic patterns* can be identified that covers the bulk of all commonly applied functionality. Any operation that maps onto the functionality as provided by such pattern is obtained by instantiating it with the proper parameters, including the function to be applied to the individual data elements. The following patterns are available in Horus: (1) *unary pixel operation*, e.g. negation, absolute value, (2) *binary pixel operation*, e.g. addition, threshold, (3) *global reduction*, e.g. sum, maximum, (4) *neighborhood operation*, e.g. percentile, median, (5) *generalized convolution*, e.g. erosion, gauss, and (6) *geometric transformations*, e.g. rotation, scaling. Recently, additional and commonly used patterns have been introduced, including iterative and recursive neighborhood operations. Current developments include patterns for operations on large datasets, as well as patterns on increasingly important data structures, such as feature vectors obtained from earlier calculations on image and video data.

For reasons of efficiency, all Parallel-Horus operations are capable of adapting to the performance characteristics of a parallel machine at hand, i.e. by being flexible in the partitioning of data structures [21], [18]. Moreover, it was realized that it is not sufficient to consider parallelization of library operations *in isolation*. Therefore, the library was extended with a run-time approach for communication minimization (called *lazy parallelization*), which automatically parallelizes a fully sequential program at runtime by inserting communication primitives and additional memory management operations whenever necessary. Results for realistic multimedia applications have shown the feasibility of the Parallel-Horus approach, with data parallel performance consistently being found to be optimal with respect to the abstraction level of message passing programs [22], [20].

Notably, Parallel-Horus was applied in the 2004 and 2005 NIST TRECVID benchmark evaluations for content-based video retrieval, and played a crucial role in achieving top-ranking results in a field of strong international competitors (incl. IBM Research and Carnegie Mellon University [20], [25]). Given this success, in combination with the observed need for heterogeneous distributed multimedia computing, we believe that the availability of a similar system for Grid environments will have an immediate, stimulating, and lasting effect on the study of the many computationally demanding MMCA problems.

3 Services-Based Multimedia Grid Computing

To arrive at a system that integrates a user transparent programming model with an efficient and easy-to-use execution model, we have combined the Parallel-Horus programming model with the now popular services-based approach to wide-area computing [1]. A services-based approach coincides well with the most common types of multimedia applications, in which distribution of tasks, as well as the parallelization thereof, is relevant (i.e., video processing, processing of large multimedia archives, parameter sweeps over multimedia data).

Specifically, we have designed a client-server based framework, that is extended with a resource broker implementation for automatic resource detection and allocation. The three logical components (i.e. client, server, and resource broker) are provided by way of three APIs that allow for services provisioning, services calling, and services registration. With these APIs, converting Parallel-Horus code to a client and corresponding server implementation is straightforward. As such, dynamic systems of distributed multimedia services, in which clients and servers can participate at will, can be created without any parallelization and distribution effort from the user. Our robot dog application discussed in Section 4 indeed is implemented as a dynamic system of this kind.

The wide-area extensions to our framework are implemented from scratch, meaning that they are not based on existing Grid middleware, or Web Services implementations [1]. Although such software systems are important for reasons of portability, and standardized protocols for data exchange, we have decided to currently rely on our own implementations — simply for reasons of efficiency and ease-of-use. As a result, we have full insight in the performance bottlenecks and potential of wide-area multimedia computing, which is often not the case when using 'black-box' Grid middleware. Having said this, comparison and integration with existing Grid solutions obviously will be an important future research issue.

4 Performance Evaluation

In this section we give an assessment of our architecture's effectiveness in providing significant performance gains. To this end, we describe the wide-area execution of a state-of-the-art vision task. Specifically, we present the application of a Sony Aibo robot dog, capable of recognizing objects, while connected to a large-scale Grid system comprising of clusters in Europe and Australia [7].

4.1 Object Recognition by a Sony Aibo Robot Dog

Our example application demonstrates object recognition performed by a Sony Aibo robot dog (see Figure 1). Irrespective of the application of a robot, the general problem of object recognition is to determine which, if any, of a given repository of objects appears in an image or video stream. It is a computationally demanding problem that involves a non-trivial tradeoff between specificity of recognition (e.g., discriminating between different faces) and invariance (e.g.,

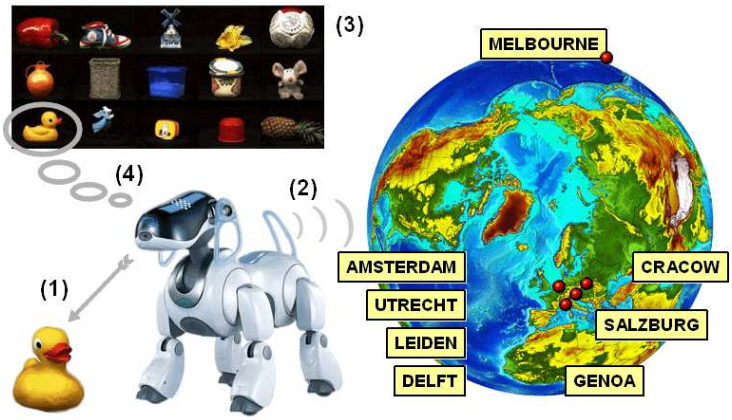


Fig. 1. *Object recognition by our robot dog: (1) an object is shown to the dog's camera; (2) video frames are processed on a per-cluster basis; (3) given the resulting feature vectors describing the scene, a database of known objects is searched; (4) in case of recognition, the dog reacts accordingly (see: www.science.uva.nl/~fjseins/aibo.html).*

to different lighting conditions). Due to the rapid increase in the size of multimedia repositories consisting of 'known' objects [7], state-of-the-art sequential computers no longer can live up to the computational demands, making high-performance distributed computing indispensable.

Local Histogram Invariants. In our robot application, we learn local histograms of invariant features for each aspect of an object. The color invariant features are highly invariant to illumination color, shadow effects, and shading of the object [6]. The features are derived from a kernel based histogram of feature responses. By exploiting natural image statistics, we model these histograms by parameterized density functions. The parameters act as a new class of photometric and geometric invariants, yielding a very condensed representation of local image content. In the learning phase of our system, a single condensed representation of each observed object is stored in a database. Subsequently, object recognition is achieved by matching local histograms extracted from the video stream generated by the camera in the dog's nose against the learned database.

In our approach, we first transform each pixel's RGB value to an opponent color representation,

$$\begin{bmatrix} E \\ E_\lambda \\ E_{\lambda\lambda} \end{bmatrix} = \begin{pmatrix} 0.06 & 0.63 & 0.27 \\ 0.3 & 0.04 & -0.35 \\ 0.34 & -0.6 & 0.17 \end{pmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad . \tag{1}$$

The rationale behind this is that the RGB sensitivity curves of the camera are transformed to Gaussian basis functions [5], being the Gaussian and its first and second order derivative. Hence, the transformed values represent an opponent color system, measuring intensity, yellow versus blue, and red versus green.

Spatial scale is incorporated by convolving the opponent color images with a Gaussian filter. Photometric invariance is now obtained by considering two non-linear transformations, as described in [5]. The first color invariant W isolates intensity variation from chromatic variation, i.e. edges due to shading, cast shadow, and albedo changes of the object surface. The second invariant feature C measures all chromatic variation in the image, disregarding intensity variation, i.e. all variation where the color of the pixels change. These invariants measure point-properties of the scene, and are referred to as *point-based invariants*.

Point-based invariants are known to be unstable and noise sensitive [29]. Increasing the scale of the Gaussian filters overcomes this partially. However, robustness is traded for invariance. A better way is to construct local histograms of responses for the color invariants. Localization is obtained by estimating the histogram under a kernel. Kernel based descriptors are known to be highly discriminative, and have been successfully applied in tracking applications [3].

Advantage of the use of an opponent color space, with additional photometric invariant transformations, is that color values are decorrelated. Hence, for a distinctive image content descriptor, we may as well use the marginal, one-dimensional, distributions for each of the color channels. This in contrast to the histogram of the full 2D chromatic or 3D color space. Hence, we use the one-dimensional channel histograms of the invariant gradients $\{W_w,\, C_{\lambda w},\, C_{\lambda\lambda w}\}$, and the edge detectors $\{W_x,\, W_y,\, C_{\lambda x},\, C_{\lambda y},\, C_{\lambda\lambda x},\, C_{\lambda\lambda y}\}$, separately.

Histogram Parameterization. From natural image statistics research, it is known that histograms of derivative filters can be well modeled by simple distributions [27]. In previous work [8], we showed that histograms of Gaussian derivative filters in a large collection of images follow a Weibull type distribution. Furthermore, the gradient magnitude for invariants W and C given above follow a Weibull distribution,

$$p(r) = \frac{1}{\beta}\left(\frac{r}{\beta}\right)^{\gamma-1} exp\left\{-\frac{1}{\gamma}\left|\frac{r}{\beta}\right|^{\gamma}\right\} \quad , \tag{2}$$

where r represents the response for one of the invariants $\{W_w,\, C_{\lambda w},\, C_{\lambda\lambda w}\}$.

The local histogram of invariants of derivative filters can be well modelled by an integrated Weibull type distribution [8],

$$p'(r) = \frac{\gamma}{2\gamma^{\frac{1}{\gamma}}\beta\Gamma\left(1/\gamma\right)} \exp\left\{-\frac{1}{\gamma}\left|\frac{r}{\beta}\right|^{\gamma}\right\} \quad . \tag{3}$$

In this case, r represents the response for one of the invariants $\{W_x,\, W_y,\, C_{\lambda x},\, C_{\lambda y},\, C_{\lambda\lambda x},\, C_{\lambda\lambda y}\}$. Furthermore, $\Gamma(\alpha)$ represents the complete Gamma function, $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1}e^{-t}dt$.

In our implementation we convert the histogram density of all local invariant histograms to Weibull form by first centering the data at the mode μ of the distribution. Then, multiplying each bin frequency $p(r_i)$ by its absolute response value $|r_i|$, and normalizing the distribution. These transformations allow the estimation of the Weibull density parameters β and γ, indicating the (local) edge contrast, and the (local) roughness or textureness, respectively [8].

Color-Based Object Recognition. In our robot dog system we apply a simple algorithm for object recognition and localization, based on the described invariant features. In the first 'learning' phase of our experiment, an object is characterized by learning the invariant Weibull parameters at fixed locations in the image, representing a sort of fixed 'retina' of receptive fields positioned at a hexagonal grid, 2σ apart, on three rings from the center of the image. Hence, a total of $1+6+12+18 = 37$ histograms are constructed. For each histogram, the invariant Weibull parameters are estimated. In the learning phase we present our dog with a set of 1,000 objects under a single visual setting. For each of these objects, the learned set of Weibull parameters is stored in a database. In the second 'recognition' phase, we validate the learning step by showing the same objects again, under many different appearances, with varying lighting direction, lighting color, and viewing position, using the same retinal structure. In this manner, our robot dog has learned each of the 1,000 objects from only one example, while being capable of recognizing more than 300 of these under a diversity of imaging conditions that may occur in everyday life. Interestingly, this recognition rate is higher than the recognition rate of around 200 objects reported for a real dog [11].

Important for this paper is the fact that (on average, see the measurements presented below) the algorithm runs at around 1 frame per 4 seconds. Moreover, we vote over the results obtained from 8 consecutive frames. As a result, object recognition takes approximately 30 seconds, which is not even close to real-time performance. As shown in the remainder of this paper, this problem is overcome by applying high-performance distributed computing at a very large scale.

4.2 Hardware Environment

Our robot dog application has been tested on multiple cluster systems located at research institutes all over the world. The bulk of the measurements were performed using the Distributed ASCI Supercomputer 2 (DAS-2), a 200-node system located at five different universities in the Netherlands [2]: Vrije Universiteit Amsterdam (VU, 72 nodes), Leiden University, University of Amsterdam, Delft University of Technology, and University of Utrecht (32 nodes each). All nodes consist of two 1-Ghz Pentium-III CPUs with up to 2 GByte of RAM, and are connected by a Myrinet-2000 network. At the time of measurement, the nodes ran RedHat Enterprise Linux AS 3.2.3.

All additional measurements have been performed using four different cluster systems in Europe and one in Australia. The first system is the 272×2 Lisa cluster located at SARA, Amsterdam, The Netherlands. All nodes consist of a 3.4-GHz Pentium Xeon CPU, with 2 GByte of RAM, are connected by a 800 Mb/sec Infiniband network, and run Debian Linux 3.1. The second system is the 36×2 Gaisberg cluster located at the Department of Scientific Computing, Salzburg University, Austria. All nodes consist of two AMD Athlon MP2800+ CPUs, with 2 GByte of RAM, are connected by a 6×6 Dolphin SGI torus network, and run the RedHat Linux 9 operating system. The third system is the 16-node cluster located at the Consiglio Nazionale delle Ricerche in Genoa, Italy.

All nodes in this system consist of a single 2.66-Ghz Pentium 4 Xeon CPU, with 0.5 GByte of RAM, are connected by a Gigabit Ethernet network, and run RedHat Enterprise Linux AS 3.2.3. The fourth system is the 8×2 Zeus cluster located at CYFRONET AGH, University of Science and Technology, Krakow, Poland. All nodes in this system consist of two 2.4-GHz Pentium 4 Xeon CPUs, with 1 GByte of RAM, and run the RedHat Linux 7.3 operating system. The final system is the 50-node Mahar cluster located at the School of Computer Science and Software Engineering, Monash University, Melbourne, Australia. All nodes consist of a single 3.0-GHz Pentium 4, with 1 GByte of RAM, are connected by a Gigabit GrangeNet network, and run Debian Linux 3.3.5.

4.3 Measurements

The sequential (server-side) Parallel-Horus implementation of our robot dog application immediately constitutes a parallel program that executes efficiently on a cluster system. Figure 2 shows measurements obtained for using each of the five DAS-2 clusters, one at a time. Results are given for the time spent on the processing of a single frame, measure both at the server side and the client application. It can be seen that the wide-area communication overhead is around 10 ms at all times, clearly an acceptable overhead. In part this is because we send frame data in JPEG compressed form — i.e. around 10-20 Kbytes per server call. Based on these results, we conclude that our framework is very effective when using the DAS-2 clusters.

In Figure 3 similar results are presented for the additional clusters. It can be seen that the wide-area overhead depends on the location of each cluster. Still, this overhead is acceptable, even in the case of Krakow and Melbourne. Interestingly, however, the application is not always very effective in obtaining high performance on all clusters. This is due to non-optimal configurations of the interconnection network (and its related software), and to high network traffic caused by other users.

Figure 4(a) shows the speedup characteristics for the application as obtained when using the DAS-2 cluster at the Vrije Universiteit in Amsterdam. We show

	VU		Leiden		UvA		Delft		Utrecht	
# CPUS	server	client	server	client	server	client	server	client	server	client
1	7.62	7.63	7.98	7.99	7.75	7.76	7.80	7.81	7.92	7.93
2	4.11	4.12	4.25	4.28	4.29	4.30	4.15	4.17	4.33	4.34
4	2.29	2.31	2.31	2.33	2.28	2.29	2.27	2.28	2.38	2.39
8	1.29	1.30	1.28	1.29	1.30	1.32	1.30	1.31	1.31	1.32
16	0.73	0.74	0.73	0.75	0.73	0.74	0.74	0.75	0.73	0.74
24	0.69	0.70	0.69	0.70	0.69	0.70	0.70	0.71	0.69	0.70
32	0.53	0.54								
40	0.60	0.61								
48	0.57	0.58								
56	0.55	0.56								
64	0.47	0.48								

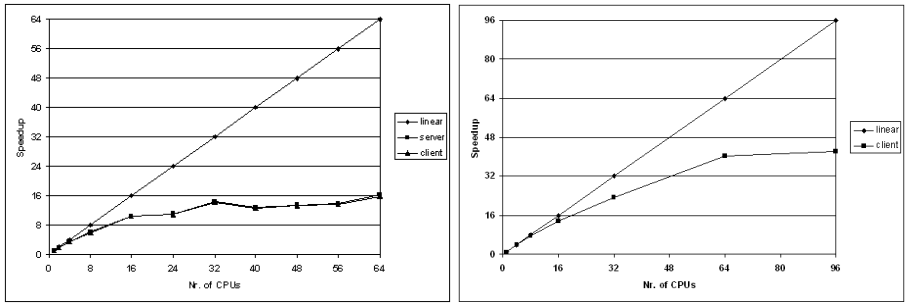
Fig. 2. *Client- and server-side results for the Aibo robot dog application using input images of size 412×318 (3-byte) pixels. Measurements for all DAS-2 clusters. Average execution times given in seconds for one frame of video. Client located at the University of Amsterdam. Resource broker located at CYFRONET AGH, Krakow, Poland.*

	SARA		SALZBURG		GENOA		KRAKOW		MELBOURNE	
# CPUS	server	client	server	client	server	client	server	client	server	client
1	1.71	1.73	2.45	2.52	2.36	2.41	4.37	4.53	9.25	9.91
2	1.01	1.03	1.29	1.36	1.31	1.36	2.75	2.91	5.37	6.02
4	0.62	0.64	0.71	0.78	0.91	0.96	2.13	2.29	3.85	4.50
8	0.46	0.48	0.41	0.47	0.76	0.81	2.11	2.27	3.36	4.02
16	0.41	0.43	0.29	0.36	0.73	0.78			3.31	3.97
24	0.40	0.42	0.31	0.37					4.03	4.70
32	0.37	0.39	0.24	0.31					6.18	6.84
40			0.32	0.39						
48			0.32	0.38						
56			0.32	0.38						
64			0.27	0.33						

Fig. 3. *Client- and server-side results for the robot dog application on five alternative clusters. Setup and data sizes as in Figure 2.*

the graph for it to be compared to the graph of Figure 4(b), which shows the speedup characteristics obtained when using four different DAS-2 clusters at the same time. The 'base' of the graph of Figure 4(b) is obtained by taking the execution time of our application running on a single node using a single DAS-2 cluster, averaged over all DAS-2 clusters. This average execution time, as well as the actual execution times obtained when using multiple DAS-2 clusters are shown in Figure 5. Interestingly, the speedup graph of Figure 4(b) (ranging over 96 CPUs) has an identical shape as the first part of the graph of Figure 4(a) (ranging over 24 CPUs). This indicates that there is no additional loss of performance at the client side when more than one multimedia server is applied. In other words, the obtained speedup with respect to the number of applied multimedia servers is fully linear. Here it should be noted that a similar speedup comparison is difficult to make when using the non-DAS-2 clusters, as each of these has different performance and speedup characteristics. Therefore we refrain from making such a comparison here.

Finally, a test in which we applied all available clusters resulted in a maximum rate of over 10 frames/sec, clearly indicating that real-time multimedia Grid computing is within reach. A video demonstration of our robot dog system can be found at www.science.uva.nl/˜fjseins/aibo.html.



(a) (b)

Fig. 4. *(a) Client- and server-side speedup for the VU results presented in Figure 2, and (b) Client-side speedup for the four-cluster measurements presented in Figure 5.*

# CPUS per	2 DAS CLUSTERS		4 DAS CLUSTERS	
CLUSTER	16 frames	1 frame	16 frames	1 frame
avg. single	124.80	7.80	124.80	7.80
1	62.32	3.90	31.01	1.94
2	33.24	2.08	16.92	1.06
4	18.49	1.16	9.33	0.58
8	10.38	0.65	5.34	0.33
16	5.98	0.37	3.11	0.19
24	5.71	0.36	2.95	0.18

Fig. 5. *Client-side results for the robot dog application. Measurements for combined use of two DAS-2 clusters (VU and UvA), as well as four DAS-2 clusters (i.e. also incl. Leiden and Delft). Average performance given in seconds for one as well as sixteen frames of video. Average single-node-single-cluster result taken from Figure 2.*

5 Future Work

The work described in this paper is part of a larger strive to bring the benefits of high-performance distributed computing to the multimedia community. While deploying Parallel-Horus on wide-area systems was shown to be straightforward, for the system to make effective use of available resources many fundamental research problems remain unsolved. One aim is to improve the performance of wide-area programs further by dynamic assignment of Parallel-Horus operations to Grid resources. The fundamental problem is to detect those (sequences of) operations that can be executed on other machines such that the overall execution time is minimized. Our approach to this problem will be feedback driven; running operations will be profiled to "learn" the respective execution times. Assuming identification of concurrently executable units, such units can be distributed to available machines. Once performance information is available, additional machines can be requested to the running application, or dropped dynamically.

The introduction of a virtual machine (vm) for multimedia computing is key to achieving these goals. On one hand, the vm will provide the conceptual framework for devising domain-specific optimization and parallelization techniques. On the other hand, the vm will provide a limited yet powerful set of operations on which techniques for reliable and self-tuning execution can be focused.

6 Conclusions

Parallel-Horus, developed at the University of Amsterdam, allows researchers in multimedia content analysis to implement high-performance applications as sequential C++ programs, using a carefully designed set of building block operations (called *algorithmic patterns*). In this paper we have described results obtained by matching the Parallel-Horus programming model with an execution model based on so-called *Wide-Area Multimedia Services*. The resulting framework, which integrates the efficient task parallel invocation of multimedia services with the automatic data parallel execution of these services, requires no parallelization or distribution effort from its users. In addition, our example application has shown that with our wide-area extensions to Parallel-Horus we

can greatly improve speedup characteristics obtained on single cluster systems. The distributed setup of our framework adds extra network overhead, but this tends to be marginal in comparison to the total execution times.

Considering these features, we feel that we have succeeded in designing and building a framework that integrates a user transparent programming model with an efficient and easy-to-use execution model, to support a wide range of multimedia applications. It should be noted, however, that the wide-area capabilities are still a 'proof-of-concept'. These have served well in showing the potential of applying Grid resources in state-of-the-art multimedia processing in a coordinated manner. However, there is still room for improvement, in functionality, reliability, fault tolerance, and performance. With such improvements, we feel that Parallel-Horus will continue to have an immediate and stimulating effect on the study of the many computationally demanding problems in multimedia content analysis. Our robot dog application is merely one of these.

Acknowledgements

The authors would like to thank the following people for their support, and for granting us access to their cluster systems: Prof. David Abramson, Collin Enticott (Monash University, Australia), Tony Adriaansen (CSIRO, Australia), Andreas Uhl, Ernst Forsthofer (Salzburg University, Austria), Daniele D'Agostino, Antonella Galizia (University of Genoa, Italy), Marian Bubak, Patryk Lason, Lukasz Skital (University of Science and Technology, Krakow, Poland), Willem Vermin (SARA, The Netherlands), Prof. Henri Bal, Thilo Kielmann, Kees Verstoep (Vrije Universiteit, Amsterdam, The Netherlands). The authors are grateful to Cees Snoek, and Michiel van Liempt for their efforts in providing us with state-of-the-art multimedia applications. Finally, thanks go out to Edwin Steffens and Arnoud Visser for providing us with a Sony Aibo robot dog.

References

1. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services - Concepts, Architectures and Applications*. Springer-Verlag, 2004.
2. H.E. Bal et al. The Distributed ASCI Supercomputer Project. *Operating Systems Review*, 34(4):76–96, 2000.
3. D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based Object Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):564–577, 2003.
4. D. Crookes, P.J. Morrow, and P.J. McParland. IAL: A Parallel Image Processing Programming Language. *IEE Proceedings, Part I*, 137(3):176–182, June 1990.
5. J.M. Geusebroek et al. Color Invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1338–1350, 2001.
6. J.M. Geusebroek et al. Color Constancy from Physical Principles. *Pattern Recognition Letters*, 24(11):1653–1662, 2003.
7. J.M. Geusebroek and F.J. Seinstra. Object Recognition by a Robot Dog Connected to a Wide-Area Grid System. In *Proceedings of ICME 2005*, July 2005.

8. J.M. Geusebroek and A.W.M. Smeulders. A Six-stimulus Theory for Stochastic Texture. *International Journal of Computer Vision*, 62(1/2):7–16, 2005.
9. L.H. Jamieson, E.J. Delp, C.-C. Wang, J. Li, and F.J. Weil. A Software Environment for Parallel Computer Vision. *IEEE Computer*, 25(2):73–75, 1992.
10. Z. Juhasz and D. Crookes. A PVM Implementation of a Portable Parallel Image Processing Library. In *Proceedings of EuroPVM'96*, pages 188–196, 1996.
11. J. Kaminski, J. Call, and J. Fischer. Word Learning in a Domestic Dog: Evidence for "Fast Mapping". *Science*, 304, June 2004.
12. D. Koelma et al. Horus C++ Reference, v. 1.1. Technical report, University of Amsterdam, The Netherlands, January 2002.
13. P.J. Morrow et al. Efficient Implementation of a Portable Parallel Programming Model for Image Processing. *Concurrency: Pract. Exp.*, 11:671–685, 1999.
14. M.R. Naphade et al. Extracting Semantics from Audiovisual Content: The Final Frontier in Multimedia Retrieval. *IEEE Trans. Neural Netw.*, 13(4):793–810, 2002.
15. C.M. Pancake and D. Bergmark. Do Parallel Languages Respond to the Needs of Scientific Programmers? *IEEE Computer*, 23(12):13–23, December 1990.
16. G.X. Ritter and J.N. Wilson. *Handbook of Computer Vision Algorithms in Image Algebra*. CRC Press, Inc, 1996.
17. J.M. Schopf and B. Nitzberg. Grids: The Top Ten Questions. *Scientific Programming*, 10(2):103–115, August 2002.
18. F.J. Seinstra et al. A Software Architecture for User Transparent Parallel Image Processing. *Parallel Computing*, 28(7–8):967–993, August 2002.
19. F.J. Seinstra et al. Finite State Machine-Based Optimization of Data Parallel Regular Domain Problems Applied in Low-Level Image Processing. *IEEE Transactions on Parallel and Distributed Systems*, 15(10):865–877, October 2004.
20. F.J. Seinstra et al. User Transparent Parallel Processing of the 2004 NIST TRECVID Data Set. In *Proc. IPDPS 2005*, Denver, CO, USA, April 2005.
21. F.J. Seinstra and D. Koelma. P-3PC: A Point-to-Point Communication Model for Automatic and Optimal Decomposition of Regular Domain Problems. *IEEE Transactions on Parallel and Distributed Systems*, 13(7):758–768, July 2002.
22. F.J. Seinstra and D. Koelma. User Transparency: A Fully Sequential Programming Model for Efficient Data Parallel Image Processing. *Concurrency and Computation: Practice & Experience*, 16(6):611–644, May 2004.
23. A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content Based Image Retrieval at the End of the Early Years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.
24. C.G.M. Snoek et al. Learning Rich Semantics from Produced Video. *ACM Transactions on Multimedia Computing, Communications and Applications*, May 2006.
25. C.G.M. Snoek and M. Worring. Multimodal Video Indexing: A Review of the State-of-the-art. *Multimedia Tools and Applications*, 25(1):January, 2005.
26. J.M. Squyres, A. Lumsdaine, and R.L. Stevenson. A Toolkit for Parallel Image Processing. In *Proc. SPIE*, San Diego, CA, USA, July 1998.
27. A. Srivastava et al. On Advances in Statistical Modeling of Natural Images. *Journal of Mathematical Imaging and Vision*, 18:17–33, 2003.
28. R. Taniguchi et al. Software Platform for Parallel Image Processing and Computer Vision. In *Proc. SPIE*, pages 2–10, San Diego, CA, USA, July 1997.
29. J. van de Weijer, T. Gevers, and J.M. Geusebroek. Color Edge and Corner Detection by Photometric Quasi-invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):625–630, 2005.
30. Q. Xiao et al. Using AVIRIS Data and Multiple-Masking Techniques to Identify and Map Urban Tree Species. *Journal of Remote Sensing*, 25(24):5637–5654, 2004.