ESC-TR-2005-062

Project Report
PCA-IRT-7

Morph Scenarios for the Integrated Radar-Tracker

J.M. Lebak
W.G. Coate

10 August 2005

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

Prepared for the Defense Advanced Research Projects Agency under
Air Force Contract FA8721-05-C-0002.

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Defense Advanced Research Projects Agency, ITO, under Air Force Contract FA8721-05-C-0002.

Massachusetts Institute of Technology
Lincoln Laboratory

Morph Scenarios for the Integrated Radar-Tracker

J.M. Lebak
W.G. Coate
Group 102

Project Report PCA-IRT-7

10 August 2005

Lexington Massachusetts

ABSTRACT

The DARPA-sponsored Polymorphous Computing Architectures (PCA) program is developing advanced computer architectures that have the capacity to adapt, or *morph*, to obtain better performance on specific problems. The key to the success of this program is the proper development of the morphing concept. One of MIT Lincoln Laboratory's contributions to this effort is the Integrated Radar-Tracker (IRT) application. The IRT consists of a Ground Moving Target Indicator (GMTI) radar and a Feature-Aided Tracker (FAT). In this document, we describe ways that the morphing capabilities of PCAs could be used by the IRT.

TABLE OF CONTENTS

List of Illustrations iv

List of Tables iv

1. Introduction 1
   - 1.1 Review of the IRT 1
   - 1.2 Review of Morphing 4
   - 1.3 Overview of Morph Scenarios in this Document 4

2. IRT Morph Scenario 5
   - 2.1 Computational Components 5
   - 2.2 Mapping the Full IRT Morph Scenario to PCAs 7
   - 2.3 Measuring Performance 8
   - 2.4 A Further Morph Scenario 9

3. Feature-Aided Tracker Morph Scenario 10
   - 3.1 Target Density Morph Change 10
   - 3.2 Parameter Mode Morph Change 10
   - 3.3 Database Morph Change 10
   - 3.4 Mean-Square Error Calculation Morph Change 11

4. Summary 12

REFERENCES 13

LIST OF ILLUSTRATIONS

Figure No.		Page
1	GMTI processing chain.	2
2	Feature-aided tracker block diagram.	3
3	Description of the Tracker Component of the Full IRT Morph Scenario.	6

LIST OF TABLES

Table No.		Page
1	Baseline Workloads for the Radar and Tracker Components.	7
2	Configurations and Workloads for the Full IRT morph scenario.	9

1. Introduction

The DARPA-sponsored Polymorphous Computing Architectures (PCA) program is developing advanced computer architectures that have the capacity to adapt, or *morph*, to obtain better performance on specific problems. The key to the success of this effort is the proper development of the morphing concept. One of MIT Lincoln Laboratory's contributions to the PCA effort is the Integrated Radar-Tracker (IRT) application. The IRT consists of a Ground Moving Target Indicator (GMTI) radar and a Feature-Aided Tracker (FAT). This application is meant to serve as an example of the types of application that are of interest for PCA. It is described in a series of technical reports [8, 9, 2, 3].

In a previous document, MIT/LL described three scenarios showing ways that the GMTI portion of the IRT could make use of the morphing capability of PCAs. The first scenario consisted of a change in the number of targets and the distribution of targets being processed. The second scenario included a change from regular, stream-based processing to data-dependent, thread-based processing. The third scenario consisted of a change in the parameter set being used for stream processing [4].

In this document, we describe additional morph scenarios, that is, ways in which the full IRT, consisting of both the radar and the tracker, could make use of morphing. The primary scenario we envision is a change from GMTI processing to FAT processing within the same hardware. We describe a simplified implementation of this scenario, methods for mapping the scenario onto PCA hardware, and ways to measure the cost and benefit of morphing in this scenario. We also describe morph scenarios for the tracker. These scenarios include parameter and target density changes for FAT, as well as changing the database used to classify targets and using morphing.

1.1 Review of the IRT

In this section we briefly review the stages and functionality of the IRT. The radar portion of the IRT is described in Section 1.1.1, and the tracker portion is described in Section 1.1.2.

1.1.1 GMTI

The GMTI component of the IRT takes unprocessed radar data and produces a set of target reports. GMTI processing is composed of the stages shown in Figure 1, which are numbered for ease of reference. Steps 1 and 3, Time Delay and Equalization and Pulse Compression, are finite impulse-response (FIR) filters. Steps 2 and 5, Adaptive Beamforming and STAP, consist of LQ factorization, backward and forward substitution, and matrix multiplication. Step 4, Doppler Filtering, is essentially a fast Fourier transform (FFT). Step 6, Detection, consists of a Constant False-Alarm Rate (CFAR) thresholding operation and three-dimensional grouping (removing duplicate detections by only considering local maxima). Step 7, Estimation, incorporates spline interpolation and maximum likelihood estimation. For more details on these stages, please see the narrowband GMTI description [9].

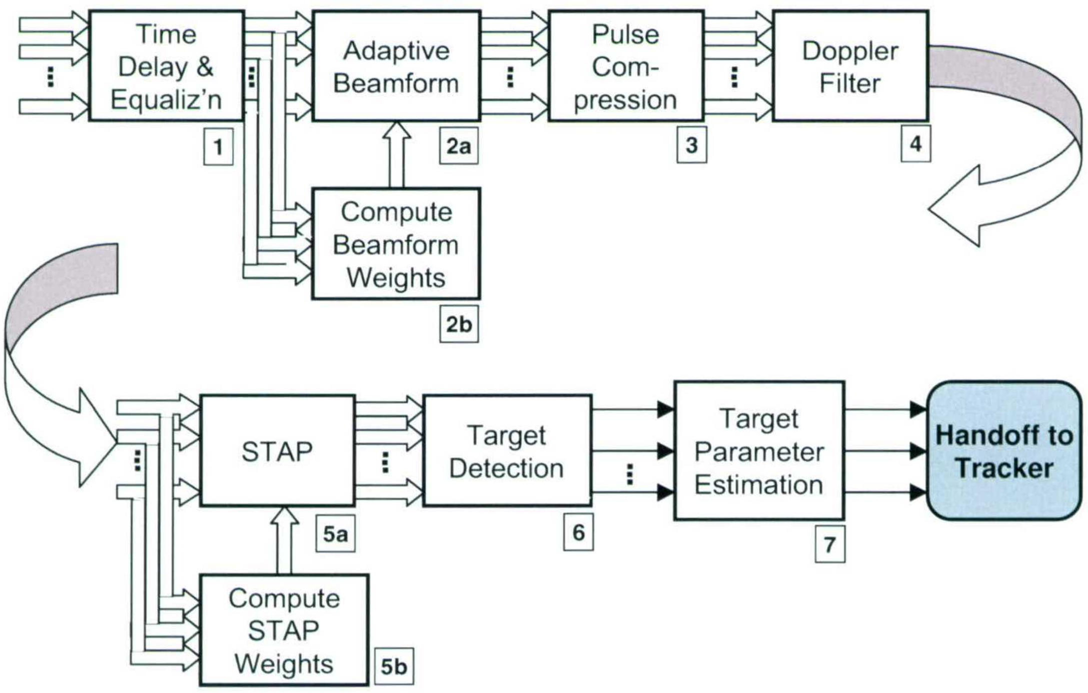

Figure 1. GMTI processing chain.

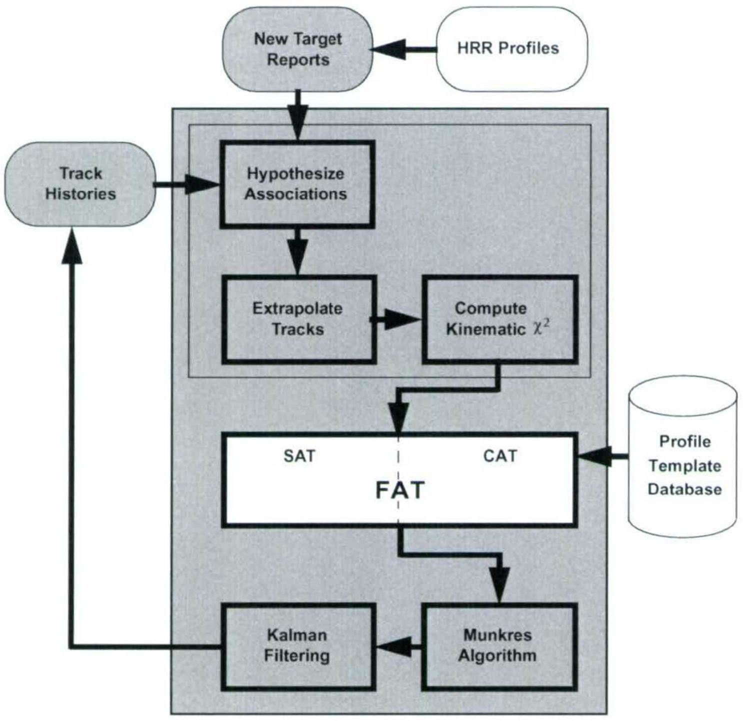

Figure 2. Feature-aided tracker block diagram.

1.1.2 FAT

The IRT's tracker consists of a standard kinematic tracker with a feature-aided tracking (FAT) capability. It operates in a series of cycles. On each cycle, it takes as input the target reports from GMTI and a set of track histories, presumed to have been produced during the previous tracker cycle. The target reports from GMTI have been enhanced with a set of *high range resolution* (HRR) profiles assumed to have been produced by an external sensor.

A block diagram of the tracker is shown in Figure 2. The gray sections correspond to the functionality of a standard kinematic tracker: the white sections correspond to the FAT capability. As Figure 2 shows, feature-aided tracking computations are inserted into the processing stream associated with the kinematic tracker.

The tracker extrapolates track history information forward to form a set of hypotheses about the target reports that should be associated with each track history. For each hypothesis, the tracker computes a likelihood value, called a *chi-squared* (χ^2) value, that reflects confidence in the hypothesis. The kinematic portion of the tracker computes this value based on the kinematics of the target reports and tracks, and so it is referred to as the kinematic χ^2 value.

The feature-aided tracking block computes an additional χ^2 value which is combined with the kinematic χ^2 value. As shown in Figure 2, feature-aided tracking is composed of two activities, signature-aided tracking (SAT) and classification-aided tracking (CAT). SAT matches the profile of the detection against the last profile associated with the track. CAT relies on a database of high-range resolution profiles against which it matches the profiles of the targets detected in this cycle.

Each of these activities produces a χ^2 value that enhances the tracker's overall ability to associate tracks and targets.

After each hypothesis has been assigned a χ^2 value, the Munkres algorithm is used to find a set of hypotheses which results in an optimal assignment of target reports to tracks. For each track, a Kalman filter is then used to update the tracks and compute a new position and velocity based on the history and the assigned target report. More details of the kinematic tracker and the feature-aided tracker can be found in additional reports [2, 3].

1.2 Review of Morphing

The basic idea of morphing is to change the configuration of a processor to maximize performance of a given operation. Performance can be measured in various ways, depending on the overall requirements of the system. Example performance metrics include operations per second or power used to perform the operation. This document describes examples of the ways that the IRT may use morphing. For a more complete description of morphing, see the documents of the PCA morphware forum [1].

For consistency with terms used to describe GMTI morph scenarios, we refer to a particular configuration of the processor as a *morph state* and the act of moving from one configuration to another as a *morph change* [4]. In this report, we will need to describe the morph state of a PCA system with r distinct resource groups. A *resource group* is an architecture-dependent and application-dependent concept. For example, the MIT Raw chip consists of 16 tiles. We could consider Raw as a set of 16 resource groups of one tile each, a set of 4 resource groups of 4 tiles each, or some other division appropriate to the application. The morph state of a PCA system divided into r resource groups will be represented by an ordered $r-$tuple $(C_1, C_2, \ldots C_r)$, where C_i is the configuration of resource set i. In general the morph state is a function of time, and changes in response to the changing needs of the application.

1.3 Overview of Morph Scenarios in this Document

The purpose of a morph scenario is to illustrate ways that an application such as the IRT could make good use of morphing. To that end, each of the morph scenarios listed here describes a workload that changes from cycle to cycle and points out how morphing could be used to adapt to this changing workload. In this document, we describe details of a full IRT morph scenario, and general parameters of four tracker morph scenarios.

The IRT morph scenario described in Chapter 2 provides details of the way in which the workload changes, similar to the details of the GMTI morph scenarios [4]. We greatly simplify the IRT processing chain to allow easy implementation of this scenario on PCAs, and we describe methods for mapping this scenario onto PCAs. Finally, we suggest concrete measurements that quantify the benefit and cost of morphing in this scenario.

The tracker morph scenarios in Chapter 3 are described in more general terms than the IRT morph scenario. These scenarios are very similar to those associated with GMTI and so our discussion here is limited to pointing out interesting ways that the tracker might use morphing and the challenges involved in doing so.

2. IRT Morph Scenario

In this chapter, we define a morph scenario for the full IRT, that is, both the radar and tracker components. This morph scenario is a compact subset of the IRT that can be used as a benchmark for morphing. It defines computationally relevant subsets of the IRT that would need to perform well for the overall application to perform well, and a changing workload that adjusts the balance between the radar and the tracker.

The amount of resources used by an application such as the IRT – that is, its *mapping* to a PCA chip – is dependent on the workload of the operations performed, the real-time requirement of the operations, and the speed at which the PCA can perform those operations. To allow the scenario definition to be independent of the particular PCA, we describe in Section 2.1 the operations performed in the scenario. We describe the mapping of those operations onto PCAs in Section 2.2. In Section 2.3, we define the measurements that are of interest for this benchmark, and in Section 2.4, we describe additional, related experiments that would tell more about the PCA under test.

2.1 Computational Components

The IRT is a large application with many different stages. To facilitate implementation of the morph scenario on PCAs, we define two components that are extremely simple and that are, respectively, computationally significant subsets of the radar and the tracker. The radar component consists of the beamforming operation from the space-time adaptive processing (STAP) phase of GMTI (Step 5a in Figure 1 on page 2). The tracker component consists of the pattern-matching operation from the classification-aided tracking (CAT) phase of the tracker. We assume that other phases of the computation are mapped onto other parts of a larger system. The computational components are described in Sections 2.1.1 and 2.1.2. The tracker component description is longer because the operations involved are less familiar.

2.1.1 Radar Component

The radar component consists of beamforming, which is a series of matrix multiply operations involving complex matrices. To simplify the computation for the benchmark, we assume that only a single beam is being formed. Given a number of channels M_R, a number of range gates N_R, and a number of Dopplers K_R, the radar component is defined as a set of K_R multiplies of a size $N_R \times M_R$ matrix with a vector of size M_R. The workload of the radar component W_R is therefore simply

$$W_R = 8M_RN_RK_R.$$

2.1.2 Tracker Component

The tracker component corresponds to the function used to calculate mean-square error (MSE), `calculateMSE()`. This function is the pattern match kernel, one of the PCA kernel benchmarks [7]: it is also described in Section 2.2 of the feature-aided tracker report [3]. It is used by the classification-aided tracker to provide an additional χ^2 value (see Figure 2 in Section 1.1.2).

```
for each of $K_T$ associations
    for each of $N_T$ templates
        for each of $\lceil M_T/3 \rceil$ shift values
            calculate MSE value (equation (1))
        Choose shift value with smallest MSE
        for each of $G$ gain values
            calculate MSE value (equation (1))
        Choose gain value with the smallest MSE
```

Figure 3. Description of the Tracker Component of the Full IRT Morph Scenario.

The mean-square error ϵ is a metric used to determine the degree to which two patterns a and t match. It may be calculated as

$$\epsilon = \frac{\sum_{k=1}^{N} \left(w_k * (a_k - t_k)^2 \right)}{\sum_{k=1}^{N} w_k}, \tag{1}$$

where w_k, $k = 1, 2, \ldots, N$ is a vector of weights. The optimal weights for the feature-aided tracker have been computed empirically. In this morph scenario, we assume a generic weighting vector is being used.

Matlab pseudo-code for the tracker component, adapted from the feature-aided tracker report [3], is shown in Figure 3. Notes on optimizing this function on a parallel processor are provided in Section 4 of the feature-aided tracker report: we will not dwell on these details here.

In Figure 3, we have made one simplification for the sake of the benchmark. In the real feature-aided tracking application, each association would need to be matched against a set of aspect angles based on the assumed aspect angle of the association. In the benchmark, we simplify the data sets involved by reusing the same set of aspect angles for each association. For more details on the original operations, please see the tracker report [3].

For the tracker component in this scenario, we use a G value of 13. Given this value of G, a pattern length M_T, a set of N_T templates, and a set of K_T associations, the workload of the tracker component can be written as

$$W_T = K_T N_T M_T \left(73 + \frac{7M_T}{3} \right).$$

2.1.3 Baseline Workload Values

Having defined the computation performed and the workload values for the radar and tracker components, we can now define specific parameter values that create a *baseline workload* for each component. The parameters that constitute a baseline workload for the radar and the tracker are summarized in Table 1. The parameters are chosen so that the workload for the radar component and for the tracker component are each approximately 1.6 Mflop.

Table 1.

Baseline Workloads for the Radar and Tracker Components.

Component	Parameter Name	Description	Value
Radar	M_R	Channels	16
	N_R	Range gates	800
	K_R	Dopplers	16
Tracker	M_T	Pattern length	32
	N_T	Number of templates	20
	K_T	Number of associations	16

2.2 Mapping the Full IRT Morph Scenario to PCAs

In this section, we describe how to map the radar and tracker components of the full IRT morph scenario onto different PCAs. The scenario consists of three *cycles* that must be executed, each with a different set of operating parameters. The first cycle, called the *balanced cycle*, uses half of the PCA to execute the radar component and half of the PCA to execute the tracker component. The second cycle, called the *radar-only* cycle, uses the entire PCA to perform the radar component. The third cycle, called the *tracker-only* cycle, uses the entire PCA to perform the tracker component.

We assume that a PCA consists of $r > 1$ resources, and divide the resources into 2 groups of $r/2$ resources: for example, 8 tiles of a 16-tile Raw chip would constitute one resource group. Each resource group can be configured into either a "radar configuration," C_R, that is more efficient for the radar component, or a "tracker configuration," C_T, that is more efficient for the tracker component. We define that a group of $r/2$ resources obtains a throughput T_R on the radar component when in configuration C_R. Similarly, define T_T to be the throughput that a group of $r/2$ resources obtains on the tracker component when in configuration C_T. These throughput values, T_R and T_T, must be obtained by measurement. For either component $i \in \{T, R\}$, define the component latency as $L_i = W_i/T_i$.

The process of mapping the full IRT morph scenario on a PCA consists of three steps.

1. Measure the value of T_R for the baseline radar workload and T_T for the baseline tracker workload.

2. Adjust the parameter values of the baseline workload so that the latencies L_R and L_T are approximately equal on the given PCA. These parameter values constitute the balanced cycle.

3. Adjust the parameter values of the balanced cycle workload to obtain the parameter values of the radar-only and tracker-only cycles.

These steps are specific to a given PCA, and reflect the load balancing that would occur in the mapping of a real application to a PCA. The steps are described in more detail below.

The first step in the mapping process, obtaining throughput values for a given implementation of the radar and tracker components, is very straightforward. Simply measure the latency L_i of the baseline workload for component i and compute

$$T_i = \frac{W_i}{L_i} = \frac{1.6 \times 10^6}{L_i}.$$

In step 2 of the mapping process, after the throughput values have been obtained, parameters of the workload are adjusted so that $L_R \approx L_T$. The two parameters we adjust are N_R and K_T: the adjusted parameter values constitute the *balanced cycle* parameters. If $T_R > T_T$, then the architecture is performing more efficiently on the radar component. We therefore increase the workload of the radar component to make the two latencies equal. Specifically, we adjust the number of range gates in the radar component by setting

$$\hat{N}_R = \frac{T_R N_R}{T_T}. \tag{2}$$

In this case we also define $\hat{K}_T = K_T$. That is, the value of the number of associations for the balanced cycle is the value from the baseline workload.

Similarly, if $T_T > T_R$, the architecture is performing more efficiently on the tracker component. In this case, we increase the workload of the tracker component to make the two latencies equal. The specific parameter adjusted is the number of associations: we set

$$\hat{K}_T = \frac{T_T K_T}{T_R}. \tag{3}$$

In this case we also define $\hat{N}_R = N_R$, that is, the number of range gates for the balanced cycle is the value from the baseline workload.

In step 3 of the mapping process we obtain the parameters for the *radar-only* and *tracker-only* cycles. In the radar-only cycle, the entire chip is being used to process the radar component. Thus we increase the radar component workload to twice the amount used in the balanced cycle, that is, we give the chip $2\hat{N}_R$ range gates to work on. Similarly, in the tracker-only cycle, the entire chip is being used to process the tracker component. Thus we give the chip $2\hat{K}_T$ associations to perform.

To summarize, the scenario consists of three cycles that must be executed. Each cycle consists of a different balance of radar uses between radar and tracker. In the balanced cycle, the workload is balanced between the radar and the tracker, and the PCA is given the configuration (C_R, C_T): that is, half of the chip is working together on the "radar" component and half is working on the "tracker" component. In the radar-only cycle, we put the chip into a configuration (C_R, C_R) and set the number of range gates in the radar component to $2\hat{N}_R$ range gates. Similarly, in the tracker-only cycle, we put the chip into a configuration (C_T, C_T) and set the number of associations in the tracker component to $2\hat{K}_T$. The configurations and workloads used in each of the cycles are summarized in Table 2.

2.3 Measuring Performance

The benefit of using a PCA in the full IRT morph scenario is assumed to be its ability to reconfigure to execute each component of the scenario well. The workload (operation count) of the operations in cycle 1 has been balanced so that each portion executes with approximately the same latency, L. The ratio of the tracker throughput T_T to radar throughput T_R on this cycle therefore gives an interesting figure of merit about the "balance" that the system is able to achieve. For an ideal system, the ratio of T_T to T_R should be close to 1. If the PCA is much better at radar-style processing than tracker-style processing, then the ratio will be greater than 1, and vice

Table 2.

Configurations and Workloads for the Full IRT morph scenario.

Cycle Number, i	Cycle Type	Configuration: Resource Group 1	Configuration: Resource Group 2	Data Size Parameter: Radar Range Gates, N_R	Data Size Parameter: Tracker Associations, K_T
1	Balanced	C_R	C_T	$\hat{N}_R$	$\hat{K}_T$
2	Radar-Only	C_R	C_R	$2\hat{N}_R$	0
3	Tracker-Only	C_T	C_T	0	$2\hat{K}_T$

versa. This ratio is related to the stability metric defined by Kuck [5] and discussed in the kernel benchmark definition report [7].

Consider the PowerPC G4 whose performance on kernel benchmarks was described in a previous report [6]. That chip obtained an average throughput on QR factorization of about 600 Mflop/s and an average throughput on pattern matching of about 110 Mflop/s. If we assume that the QR factorization throughput is indicative of the matrix multiply throughput then this would lead us to believe that the ratio for a PowerPC G4 for this benchmark would be 0.18 or less. A goal for PCAs would be to exceed the value of this ratio for conventional architectures.

Another important metric to be observed from this scenario is the total latency to execute the three cycles compared to the base latency, L, to execute any one of the three cycles. If the workload is truly balanced and a morph change takes zero time, then the total latency will be $3L$. The latency may be greater than $3L$ because of the time necessary to implement the morph change, that is, to reconfigure the processor resources. The latency may also be greater because of inefficiencies in the sharing of problems among resources. This scenario can therefore be used to demonstrate the overhead of morphing on a particular PCA. This overhead may include aspects of both hardware and software overhead.

2.4 A Further Morph Scenario

An interesting exercise if the PCA consists of $r \geq 4$ resources would be to change the mix of workloads from 50% radar and 50% tracker to 75% radar and 25% tracker and then to 25% radar and 75% tracker. To handle this, the resources would be configured as, respectively, (C_R, C_R, C_T, C_T) for the first workload, (C_R, C_R, C_R, C_T) for the second workload, and (C_R, C_T, C_T, C_T) for the third workload. This could be implemented, for example, on the Raw processor by configuring four of the 16 tiles as a unit. The advantage of this scenario over the original scenario is that each cycle includes some radar processing and some tracker processing.

3. Feature-Aided Tracker Morph Scenario

In this chapter, we briefly describe additional morph scenarios for the feature-aided tracker. These scenarios are very similar to the scenarios previously described for GMTI [4]. Their purpose is to bring up additional issues that will need to be considered in implementing applications using PCAs.

3.1 Target Density Morph Change

GMTI morph scenario "A" explored a change in the number of targets in a given area from a low number to a high number. This caused the portion of the PCA performing target parameter estimation to have to cope with a data-dependent load balancing problem. A similar problem could be devised for the feature-aided tracker. In this case, the entire tracker would need to be re-balanced.

The tracker can be parallelized either by distributing the detections coming out of GMTI or by distributing the tracks from the previous tracker cycle. If the parallelism comes from distributing the tracks, then the distribution, while data-dependent, can be determined ahead of the arrival of detections. Such a scheme would allow the latency of reconfiguring the chip to efficiently distribute the data to be hidden from the overall latency to process the tracks.

3.2 Parameter Mode Morph Change

GMTI morph scenario "C" explores a change in parameters in GMTI from one cycle to the next. A similar morph change could be defined for FAT. However, because the tracker maintains state information and GMTI does not, implementing this morph change for the tracker imposes additional challenges for the system designer.

If the state is to change whenever parameters are switched, then the PCA system must provide a way to save state information before the morph change and to restore it after the morph change. This is very similar to standard operating system mechanisms used to implement multi-tasking or multi-threading.

A harder problem is to somehow preserve the state while switching parameters. This is really a challenge for the application designer. For example, if the area under surveillance changes size, or the pattern length changes to reflect a different resolution, the application would have to know how to handle this. The PCA system might be able to provide tools for the application designer to deal with these cases. However, assuring correct behavior in these cases is beyond the scope of the PCA program.

3.3 Database Morph Change

FAT makes use of a pre-computed database of templates. These templates are chosen to reflect the targets that the platform will see during its mission. If we assume that the platform on which the tracker resides moves from one geographic region to another during the mission, then the template database might change during the mission.

The FAT database includes statistics about how well patterns in the database match other patterns in the database. In the current implementation of FAT, generating a database takes a very long time due to the need to generate these statistics. Therefore, it is logical to assume that these databases would be generated ahead of time, and the feature-aided tracker merely switches between them during the mission. This is therefore very similar to GMTI morph scenario "C" and so it is not described in detail in this document.

3.4 Mean-Square Error Calculation Morph Change

One unique opportunity presented by the morphing capability of PCAs is the potential to perform data-dependent run-time optimization of a calculation. An example of this occurs in the mean-square error (MSE) calculation of the feature-aided tracker. As has previously been pointed out, the MSE calculation is performed many times. It is performed for each association between a track and a target, for each template in the library, and for each aspect angle in a range around the assumed aspect angle of the track-target pair.

If a particular target is associated with two or more tracks, and the aspect angles of the tracks are similar, then many redundant MSE calculations will be performed. Designing the application to understand where these redundant calculations occur and can be optimized is a significant challenge. The amount of overlap is not known until run-time, so any optimization cannot be performed until then. However, performing such optimization could allow the system to dynamically adjust its resource use. The possibility of performing such optimization leads one to ask what hardware mechanisms the PCA system could provide to allow such optimization to be done.

4. Summary

We have described in detail a morph scenario reflecting a subset of the PCA integrated-radar tracker. The application subset we describe consists of a radar component and a tracker component. Each component is designed to be small enough to implement easily as well as scalable to match the specific PCA under test. We have described a workload mix for the application subset that tests the ability of the PCA under test to reconfigure for different situations. We have also described additional ways that the IRT's feature-aided tracker could use the morphing capabilities of PCAs. These scenarios can serve to focus additional discussion of morphing in the PCA community.

REFERENCES

1. Dan Campbell, Dennis Cottel, Randall Judd, and Mark Richards. Introduction to morphware: Software architecture for polymorphous computing architectures. Technical report, Georgia Institute of Technology, 2004.

2. W. Coate. Preliminary design review: Kinematic tracking for the PCA integrated radar-tracker application. Project Report PCA-IRT-4, MIT Lincoln Laboratory, Lexington, MA, February 2003.

3. W. Coate and M. Arakawa. Preliminary design review: Feature-aided tracking for the PCA integrated radar-tracker application. Project Report PCA-IRT-5, MIT Lincoln Laboratory, Lexington, MA, October 2004.

4. W. Coate and J. Lebak. Morphing scenarios for the GMTI portion of the PCA integrated radar-tracker. Project Report PCA-IRT-6, MIT Lincoln Laboratory, Lexington, MA, February 2004.

5. David J. Kuck. *High Performance Computing: Challenges for Future Systems.* Oxford University Press, New York, NY, 1996.

6. James Lebak, Hector Chan, Ryan Haney, and Edmund Wong. Polymorphous computing architectures (PCA) kernel benchmark measurements on the PowerPC G4. Project Report PCA-Kernel-2, MIT Lincoln Laboratory, Lexington, MA, January 2004.

7. James Lebak, Albert Reuther, and Edmund Wong. Polymorphous computing architectures (PCA) kernel-level benchmarks. Project Report PCA-Kernel-1, MIT Lincoln Laboratory, Lexington, MA, January 2004.

8. James M. Lebak. Preliminary design review: PCA integrated radar-tracker application. Project Report PCA-IRT-1, MIT Lincoln Laboratory, Lexington, MA, April 2002.

9. Albert I. Reuther. Preliminary design review: Narrowband GMTI processing for the basic PCA radar-tracker application. Project Report PCA-IRT-3, MIT Lincoln Laboratory, Lexington, MA, February 2003.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 10 August 2005	**2. REPORT TYPE** Project Report	**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE Morph Scenarios for the Integrated Radar-Tracker		**5a. CONTRACT NUMBER** FA8721-05-C-0002
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) J. M. Lebak and W. G. Coate		**5d. PROJECT NUMBER** 1084
		5e. TASK NUMBER 1
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) MIT Lincoln Laboratory 244 Wood Street Lexington, MA 02420-9108		**8. PERFORMING ORGANIZATION REPORT NUMBER** PR-PCA-IRT-7
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) DARPA/ITO 3701 Fairfax Drive Arlington, VA 22203-1714		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) ESC-TR-2005-062

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The DARPA-sponsored Polymorphous Computing Architectures (PCA) program is developing advanced computer architectures that have the capability to adapt, or *morph*, to obtain better performance on specific problems. The key to the success of this program is the proper development of the morphing concept. One of MIT Lincoln Laboratory's contributions to this effort is the Integrated Radar-Tracker (IRT) application. The IRT consists of a Ground Moving Target Indicator (GMTI) radar and a Feature-Aided Tracker (FAT). In this document, we describe ways that the morphing capabilities of PCAs could be used by the IRT.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT Unclassified	**b. ABSTRACT** Same as Report	**c. THIS PAGE** Same as Report	Same as Report	17	**19b. TELEPHONE NUMBER** *(include area code)*

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18