Applications of the split protocol paradigm

Article *in* International Journal of Computers and Their Applications · June 2014

CITATION
1

READS
225

5 authors, including:

Bharat S Rawal
Gannon University
120 PUBLICATIONS **289** CITATIONS
SEE PROFILE

Ramesh K Karne
Towson University
95 PUBLICATIONS **639** CITATIONS
SEE PROFILE

Patrick Appiah-Kubi
Indiana State University
14 PUBLICATIONS **81** CITATIONS
SEE PROFILE

A publication of ISCA*:
International Society for Computers
and Their Applications

INTERNATIONAL JOURNAL OF COMPUTERS AND THEIR APPLICATIONS

TABLE OF CONTENTS

*** "International Journal of Computers and Their Applications is abstracted and indexed in INSPEC and Scopus."**

Volume 21, No. 2, June 2014 **ISSN 1076-5204**

International Journal of Computers and Their Applications

A publication of the International Society for Computers and Their Applications

ISCA Headquarters···•···64 White Oak Court, Winona, MN 55987···•···Phone: (507) 458-4517
E-mail: isca@ipass.net • URL: http://www.isca-hq.org

Applications of the Split Protocol Paradigm

Bharat S. Rawal*, Ramesh K. Karne*, Alexander L. Wijesinha*,
Patrick Appiah-Kubi*, and Sonjie Liang*
Towson University, Towson, Maryland, USA

Abstract

A protocol is considered to be an *in-separable* entity between two or more communicating systems. In protocol splitting, a protocol is split into two sub-entities. The split protocol performs identical functions as the non-split protocol. Protocol splitting enables many innovations that are not possible with traditional non-split protocols. In this paper, we describe the split protocol concept and explore its use in implementing client server architectures, mini-cluster configurations, and migratory servers. Protocol splitting can be used to improve server performance, construct a variety of mini-cluster configurations to serve large workloads, build clusters without using expensive partitioning strategies, build simple migratory server systems, enrich client server architectures to accommodate client splitting, and achieve inherent server reliability. In particular, we consider the architecture, design and implementation of relevant split Web server and client applications. The primary purpose of this paper is to consolidate previous work on split protocols and discuss their broader impacts on the design and implementation of future communication system architectures.

Key Words: Bare machine computing, load balancing, server clusters, server migration, split protocols, web server performance.

1 Motivation

Network protocols enable interactions between two or more communicating entities via an exchange of messages. In many cases, two-way interactions between these entities are tightly coupled to process the messages following an event/action client-server model based on requests and replies. Examples of protocols that operate in this manner include HTTP, FTP, TLS, SMTP, and TCP. In general, such protocols are implemented on the client and server as ***in-separable*** entities.

This in-separable characteristic of protocols provides the motivation for a different approach to client-server interaction; that is, splitting a protocol so that conventional inseparability is broken. A protocol can be split in many ways as described in this paper. Splitting yields surprising results that can be useful in the design of future client-server systems, server clusters and migratory servers.

2 Introduction

We illustrate the split protocol concept by means of Web server and client applications that are designed and implemented using the bare machine computing (BMC) paradigm, which extends the dispersed operating system computing (DOSC) concept [16]. Bare PC applications are easier and simpler to use in the design and implementation of split systems. In principle, splitting could be done using conventional non-bare systems, but would be difficult to implement in the presence of an operating system (OS). In the BMC paradigm, the OS or kernel is completely eliminated and application objects (AO) [14] carry their own network stack and drivers to run independently on the bare machine. All BMC applications are self-supporting, self-managed and self-executable, thus allowing the application programmer to solely control the applications and their execution environments. Many client-server applications (for example [1, 9, 11]) have been built using bare PCs.

Web servers play a critical and central role in the Internet. Since Web servers only perform a certain (limited) set of functions and do not require elaborate user interfaces, they (and servers in general) are especially suited to be designed as bare PC applications. Furthermore, as opposed to bare PC Web servers, conventional Web servers require periodic maintenance, updates, and are prone to OS based attacks. Also, a bare PC Web server will have better performance, greater reliability, easier maintainability and more longevity than an OS-based server.

Many approaches to designing servers have been proposed in the literature. Google proposed its own lean Linux kernel [2]. Factored OS [30] targets scalability, bare-metal Linux [29] is a trimmed version of Linux, and SWILL [17] allows the addition of a simple Web server to applications. Exokernel [8] recommends moving network intensive processes from kernel to applications in order to speed-up processes, and the Fluke kernel [9] provides an API characterized by full interruptible and restartable ("atomic") kernel operations that has many advantages for applications. Tiny OS [27] suggests a small footprint for OS, OS Kit [20] provides system libraries for open systems, and IO-Lite [21] provided fast I/O calls bypassing system calls. Palacios and Kitten [18] are tools

* Department of Computer and Information Sciences. E-mail: (bsrawalkshatriya, rkarne, awijesinha, appiahkubi, sliang)@towson.edu.

designed to provide a thin layer over the hardware to support both full-based virtualization and native code bases. ChromiumOS [5] is an operating system designed for Web users. Many server systems disable the usual OS functions such as shell scripts, logs, and remote logins so they can gain performance. Commercial servers such as Apache and IIS focus their designs on optimization for increased performance and also provide an API to hardware resources instead of using system calls.

The above approaches require some OS layer, but the BMC paradigm proposes the avoidance of any OS, kernel, or centralized software to manage resources. In this paradigm, control is given back to the application programmer [16]. When a machine is made bare, it enables a programmer to develop applications that are completely independent of computing environments and only dependent on the underlying CPU architecture. However, the application programmer's job becomes more complex now, since it is necessary to deal with both application and system knowledge. On the other hand, the applications themselves are simple and easy to code once the programmer understands the underlying principles of the BMC paradigm. The design and implementation of a bare PC Web server are presented in [11], and the direct hardware interfaces for an application object (AO) are described in [15].

Web server reliability and load distribution are important problems that are still being addressed with a variety of techniques. In particular, load balancing techniques are used at various layers of the protocol stack to share the load among a group of Web servers [6]. In approaches such as Migratory TCP (M-TCP) [26], a client connection is migrated to a new server and an adaptation of TCP enables migration of the connection state. In contrast, the split technique when applied to TCP enables the splitting of a TCP connection between servers so that one server handles connection establishment and another handles data transfer. Splitting also allows a server to self-delegate a percentage of its connection requests to another server for data transfer and enables servers to share the load without a central control and without any client involvement.

To explain how a TCP connection is split by a BMC Web server, consider the message exchange in Figure 1. When an HTTP GET request arrives after the TCP connection is established, the BMC Web server sends a response (GET-ACK) to the client, and updates the client's state. The GET request is then processed by the server and the requested file is sent to the client during the data transfer. This process differs somewhat based on whether the HTTP request is static or dynamic. Although we address splitting only for static requests, the techniques apply to dynamic requests as well. Once the data transfer is complete, the connection is closed by exchanging the usual FIN and FIN-ACK messages. Although multiple GET requests can be sent using a single TCP connection, for ease of explanation, we only consider the case of a single GET per connection. A single HTTP request and its underlying TCP messages can thus be divided into connection establishment (CE), data transfer (DT), and connection termination (CT) phases.

The novel splitting concept presented here is based on splitting the HTTP request and underlying TCP connection into two sets {CE, CT} and {DT}. It allows one server to handle connections and the other to handle data without a need for too many interactions between servers. The data could reside on only one server or on both servers if reliability is desired. Splitting a client's HTTP request and the underlying TCP connection in this manner also provides the additional benefit of data location anonymity while enabling load sharing among servers. In the future, server machines optimized to handle only connection requests and others optimized to handle only data transfer could be built to take advantage of splitting.

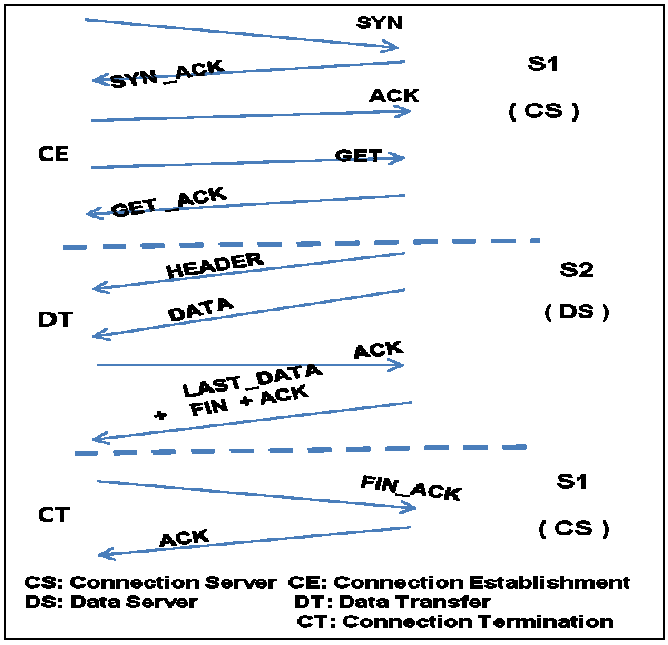


Figure 1: HTTP/TCP protocol interactions

Load balancing is frequently used to enable Web servers to dynamically share the workload. For load balancing, a wide variety of clustering server techniques [13] is employed. Most load balancing systems used in practice require a central control system such as a load balancing switch or dispatcher [28]. Splitting enables a new approach to load balancing whereby HTTP requests are split among a set of two to four servers, where one or more connection servers (CSs) handle TCP connections and may delegate a fraction (or all) requests to one or more data servers (DSs) that serve the data [22]. For example, the data transfer of a large file could be assigned to a DS and the data transfer of a small file could be handled by the CS itself. This approach is different from the approach introduced in [7] where TCP is spliced for URL-aware redirections.

As noted earlier, no client involvement is necessary for splitting unlike migratory or M-TCP [14]. In [22], splitting using a single CS and a DS was shown to improve

performance compared to non-split systems. Since the DSs are completely anonymous and invisible (they use the IP address of the delegating CS), it would be harder for attackers to access them. In particular, communication between DSs and clients is only one-way, and DSs can be configured to only respond to inter-server packets from an authenticated CS. We studied the performance of three different configurations of Web server clusters based on HTTP splitting by measuring the throughput (in requests/s) and also connection and response times at the client.

In real world applications, some servers may be close to data sources, and some servers may be close to clients. Splitting a client's HTTP request and the underlying TCP connection in this manner allows servers to dynamically balance the workload. The split concept has being tested in a LAN with multiple subnets connected by routers. In HTTP splitting, clients can be located anywhere on the Internet and interact with servers at a different location. However, there are security and other issues that require the CS and DS to be on the same LAN. These issues are discussed in more detail in [22].

The rest of the paper will describe the split protocol architecture, and the design, implementation and performance of several network applications. Since these applications were discussed in [22-25], we only consolidate and summarize the main concepts and results here.

3 Split-Protocol Server Architecture

The basic split protocol architecture used for the experiments described in [22] is reproduced here for illustration in Figure 2. A given request is split at the GET command between a CS and a DS. The CS handles the connections, and the DS handles the data transfer. In addition to connections, the CS also handles the data ACKs and the connection closing. The CS has complete knowledge of the requested file, its name, size, and other attributes, but it may or may not have the file itself. However, the DS has the file and serves the data to the client.

After the GET command is received by CS, it sends an ACK

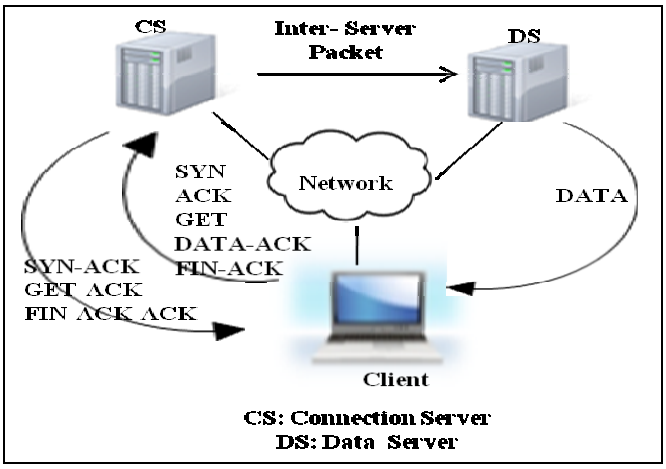


Figure 2: Split-protocol architecture

to the client and also sends a delegate message DM1 to DS. The message DM1 contains the state of the request that is stored in CS in the form of an entry in the TCP table (referred to as a TCB entry). When DM1 reaches the DS, it creates its own TCB entry and starts processing this request as if it was initiated in the DS itself. When a DS sends data to the client it uses the CS's IP. After the CS receives a FIN-ACK from the client to signify connection closing, it sends another inter-server packet referred to as DM2 to DS. The DM2 received by DS will close the state of the request in DS. These DM1 and DM2 inter server packets serve as the start and end of a request at DS. More details of the design and implementation can be found in [22].

The CS and DS architecture in Figure 2 allows for a variety of delegation configurations. A request received by CS can be either processed wholly at CS or delegated to DS. That is, some requests can be processed at CS and some can be delegated to DS resulting in a variation of the delegation ratio. As CS and DS are identical functional units, they can also perform the role of a CS or a DS. These novel splitting techniques and the associated Web server architecture could be used in distributed computing and for improving server reliability.

4 Mini-Cluster Configurations

The initial studies conducted on split servers and their performance led to the development of mini-cluster configurations. A variety of cluster configurations together with the architecture, design, implementation and performance of mini-cluster configurations were presented in [23]. Mini-cluster configurations also offer better reliability because of the interchangeable nature of CSs and DSs.

4.1 Cluster Configurations

In a split request, the client first sends the request to a CS, the CS sends an inter-server packet to a DS, and the DS sends the data packets to the client. Inter-server packets may also be sent during the data transfer phase to update the DS if retransmissions are needed. With partial delegation configuration, the CS delegates a fraction of its requests to DSs whereas in full delegation configuration, the CS delegates all its requests to DSs.

This mini Web server clustering technique uses two or more servers with protocol splitting. Configuration 1 in Figure 3 shows full delegation with one CS, one DS, and a set of clients sending requests to the CS. The DS and CS have different IP addresses, but the DS sends data to a client using the IP address of the CS. Configuration 2 in Figure 4 shows a single CS with two or more DSs in the system with partial or full delegation. In partial delegation mode, clients designated as non-split request clients (NSRCs) send requests to the CS, and these requests are processed completely by the CS as usual. The connections between the NSRCs and the CSs are shown as dotted lines. With full delegation, clients designated as split-request clients (SRCs) make requests to the CS, and these requests are delegated to DSs. For full delegation, there are no

NSRCs in the system. When requests are delegated to DSs, we assume they are equally distributed among the DSs in a round-robin fashion. It is also possible to employ other distribution strategies. Configuration 3 in Figure 5 shows two

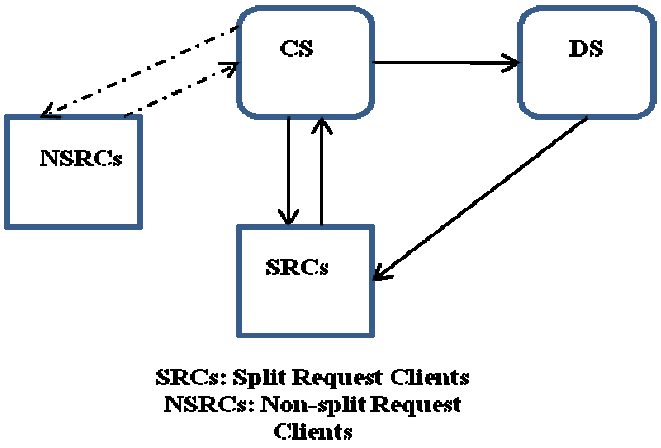


Figure 3: Split architecture configuration 1

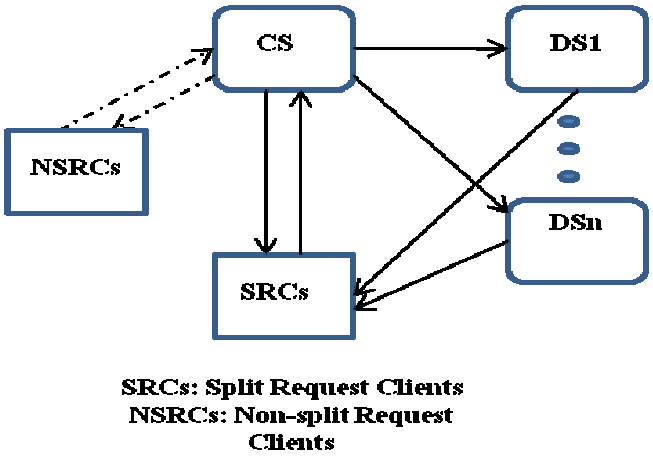


Figure 4: Split architecture configuration 2

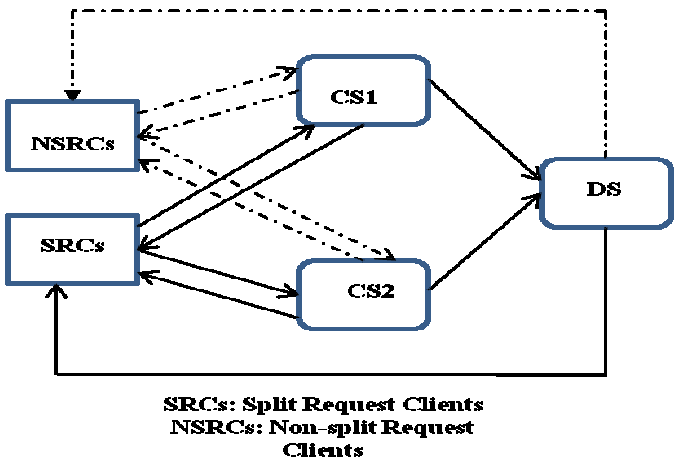


Figure 5: Split architecture configuration 3

CSs and one DS with both SRCs and NSRCs. For this configuration, we used small file sizes to avoid overloading the single DS. Although we have not done so, multiple DSs could be added as in Configuration 2.

4.2 Experimental Results

Some results of performance studies conducted on the mini-cluster configurations are given here. Figure 6 compares the throughput for split servers with full and partial delegation by varying the file size. The maximum throughput and a performance improvement of 25 percent are attained for 64 KB files with partial delegation. For 100 KB and 128 KB files, the performance improvements due to splitting are 17.7 and 12.5 percent, respectively with partial delegation. Notice that when two identical servers are used (no split), the maximum request/sec that can be achieved is about 2000 requests/sec for 64K resource file size, assuming there is no overhead for load balancing. In the case of split protocol servers however, the maximum request/sec that can be achieved is 2500 requests/sec, which is 25 percent more than theoretical maximum. This performance improvement shows the benefit of using the split protocol technique.

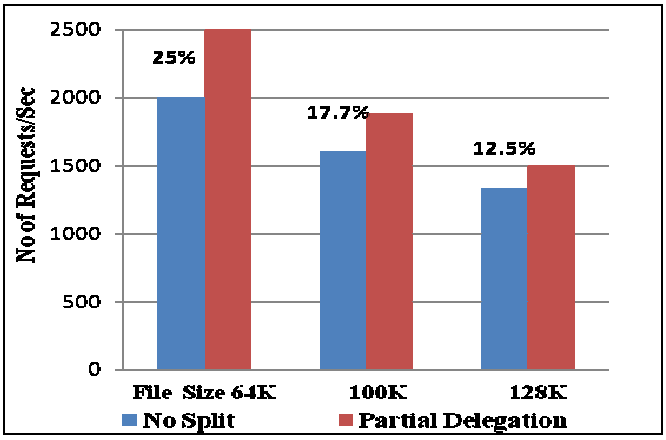


Figure 6: Throughput with full/partial delegation for varying file sizes (configuration 2)

Figure 7 shows the throughput for three servers with full and partial delegation using configuration 3. In this configuration, the throughput improvement over three non-split servers with full delegation is about 6.5 percent. However in the case of partial delegation, the throughput improvement is about 22 percent over three non-split servers.

The configuration studied here does not require a central load balancer or dispatcher, and is completely transparent to the client. The experimental results for this configuration suggests that scalable Web server clusters can be built using one or more split server systems, each consisting of 3-4 servers. More studies are also needed to evaluate the security benefits of split server clusters, and their scalability and performance with a variety of workloads.

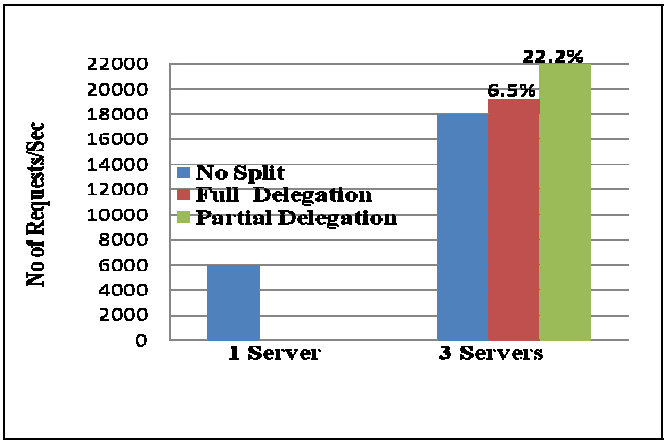


Figure 7: Throughput with full/partial delegation (configuration 3, file size 4KB)

5 Modified Client/Server Architecture

The nature of the interaction between the CS, DS and the client is evident on examining Figures 1 and 2. The CS handles all connections and it communicates with the client by sending and receiving packets. The DS only communicates with the client by sending data to it. The CS also sends an inter-server packet to DS to provide client's request and its state. The CS is connected to the client throughout its session and during the processing of a given request. In this architecture, one CS interfaces with one or more DSs to provide client services and thus becomes a bottleneck in a given mini-cluster configuration [23].

To address this problem, the client-server architecture was modified. In this new architecture [25], connections and data transfers are separated entirely as shown in Figure 8. The data servers (one or more) and their counterpart connection server can be on the same subnet or on different subnets within the same LAN. The CSs can be monitored for ongoing connections with the clients, while isolating the data servers

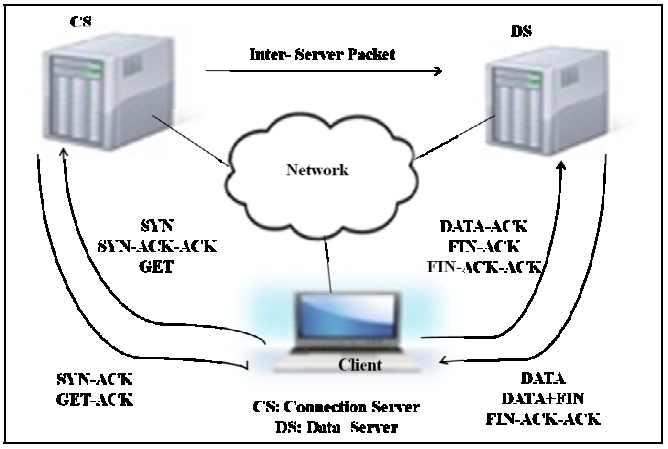


Figure 8: Split protocol architecture (modified client/server architecture)

from the clients. Splitting the connections among the CS and DS servers enables increased security to be provided at a server level. When a connection is established between a client and a CS, the CS will send an inter-server packet to a DS and terminate its connection processing; the DS will send the data and close the connection. Notice in Figure 8 that the client sends the SYN, SYN-ACK-ACK and GET to CS and CS sends SYN-ACK and GET-ACK to the client. After CS processes the connection, it sends a message to DS through an inter-server packet and eliminates this connection at CS. The rest of the connection and interactions related to DATA, ACK and FIN-ACK will be processed by DS. We have freed up CS completely after the GET is processed, but the client must be made aware of the DS as described in [25].

5.1 Configurations

Figure 9 shows a general configuration for connecting one CS, one or more DSs and one or more clients. A resource file size of 64K is used throughout our measurements. The CS will delegate all its requests to one or more DSs for data processing.

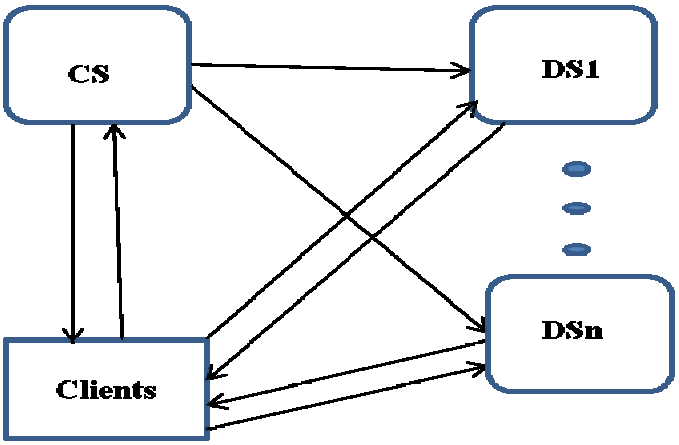


Figure 9: Split Architecture Configurations (Modified Client/Server Architecture)

5.2 Experimental Results

In this system configuration, as CS does not wait until the connection is closed, the CS utilization was very low, so more DSs can be added to the configuration to serve the clients. A linear performance improvement was achieved with up to 4 DSs added to the configuration. This linear performance gain is expected until the CS gets saturated. The linear performance is also expected because the CS causes no bottleneck and all DSs execute concurrently and independently to process client requests. The number of DS servers connected to a single CS server can be estimated to be 15, by extrapolating the observed results. This observation can be used when forming large cluster configurations with split servers. Although the Google architecture uses clusters [2], no protocol splitting is involved.

6 Server Migration

Novelties derived from the split protocol concept can be leveraged in developing server migration architectures [24]. Server migration is usually needed in catastrophic situations or to perform backups and recovery. The split protocol architecture has a minimum of two servers (CS and DS). These two servers have the same state and context of a given request. The CS has the state of a request until GET arrives and the DS has the same state of the request until the data is sent to the client. In essence, the DS acts as a shadow server for CS for a given request i.e., split protocol servers inherently support redundancy and reliability by design. There is no need for explicit shadow servers in split protocol server systems.

The bare server migration architecture proposed here is not same as process migration [19] as there is no process context in the state of a request in a bare implementation. Mobile agents, Aglets (Java based agent technology from IBM), mobile virtual desktop computing [3], and similar ideas have been investigated previously by other researchers. Mobile Web servers [4] provide Web services for mobile devices. Most of these earlier approaches are based on a computing environment where there is some sort of resident operating system. Typically, mobility is then implemented at an application level and the client may have to be aware of migration either through different code or by making a new connection to the server. None of these approaches can handle switch over within a single request.

In bare split protocol servers, the state of a given request is stored in a TCP table whose entries are referred to as TCB entries. Each TCB entry has the complete state of a given request, which consists of about 168 bytes. If this state is moved from a CS to another CS, the latter will simply resume execution from this current state. The 168 byte packet containing the TCB entry can be sent to another CS as an inter server packet with a special tag. When this special tagged packet arrives, the new CS saves the TCB entry in its own TCP table and takes over processing the request.

Migrating servers in this manner poses several networking challenges since there are no changes made to the client and the client has no knowledge of the migration process. One of the main difficulties is to deliver the client connection to the new server who needs to use the same IP as the old server when communicating with the client. This change needs to occur when the old server crashes, or it wishes to hand its connections over to the new server. Another difficulty is migrating existing requests in a CS on one network to a CS in another network. The number of requests residing in a given CS depends on the current client load and the remaining requests to be serviced. Both issues were addressed in a LAN environment by making minimal changes to the server code to handle the migration process. The following sub-sections give more details of the architecture, design, implementation and measurements relevant to migratory bare servers.

6.1 Migration Architecture

In order to implement server migration, a test LAN was set up as shown in Figure 10. The CS, DS and client operate as described in earlier sections. Section 6.3.1 provides more details of the network. The new server CS* to which CS will migrate is connected to the network, but it will not process any client requests and is in stand-by mode. Note that CS and CS* are on different subnets. The relevant IP and MAC addresses used in our case are shown in the figure. When the client sends a request to the CS, it is delegated to the DS and the DS serves the data to the client. In our network, the client request goes through Router#3 and Router#2 via Switch#0 to reach CS. The relevant routing table entry at Router#3 shows 10.55.12.0 → 10.55.10.10 since Router#2 is the next hop to reach the 10.55.12.0/24 network. Once CS* takes over, packets from the client to CS (with destination address 10. 55.12.241) will need to be sent by Router#3 to CS*, which resides on the 10.55.10.0/24 network.

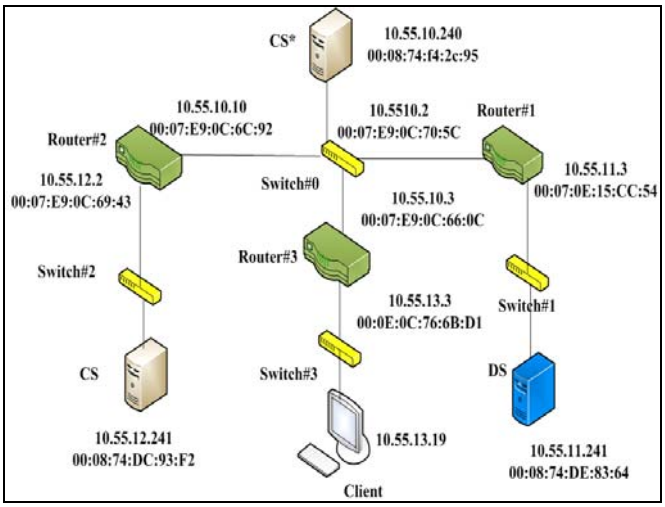


Figure 10: Test LAN for migration

6.2 Design and Implementation

As described in [24], the bare CS design is modified to enable migration capabilities in the server. When a CS is ready to migrate (as triggered by a timing point), it sends all of its pending requests to CS*, which takes over the role of CS. The CS has prior knowledge of CS* and its IP address. Prior to migration, inter-server packets are being sent from the CS to the DS as usual when each GET request arrives. Under large load conditions, it is possible that CS could have many unprocessed requests in the TCB table. In addition to these pending requests, new requests may still come in after the migration event is triggered. These requests will be lost and the client will retransmit them. The retransmitted requests will be processed by CS*. Before migration, CS sends a final inter-server packet to CS* to confirm it is handing over. After this, CS* will send an ARP broadcast to enable its MAC address to be bound to the IP address of CS. Minimal modifications were made to the current server design and inter-server packet code to implement server migration. In essence, the bare PC server split protocol design is easily adapted for migration.

An alternative configuration for migration is to have the CS send its pending requests to its DS before handing over. The DS receives the pending requests and stores them in its TCB. The DS may still have some previous data transfer requests to be processed and sent to clients, so it acts as a DS and CS at the same time until all the previous data requests are processed before turning into a CS (i.e., CS*). CSs and DSs are designed to take either a CS role, or a DS role, or both at any given point in time. This configuration is unique to the bare PC split protocol server design and can be implemented with no additional logic or complexity.

We now briefly describe the implementation details of server migration for our test LAN. Recall that CS and CS* are on different subnets (Figure 10). After CS* takes over from CS, until the new route from the client to CS* which is located at 10.55.10.240 is set up through the relevant routers, packets sent by the client to CS are not delivered. However, the client is unaware of server migration and continues to send packets using the TCP connection to CS. To set up the new route, the Linux script program in Figure 11 was written. It enables

```
# monitor routes
while [ ${_unl} -gt 0 ]
do
 if [ $( ping -c 1 ${_ip1} |grep -ic "100% packet loss" ) -gt 0 ];then
   if [ "${_interf}" = "10.55.10.10" ]; then
     if [ ${_flg0} -eq 0 ]; then
       ip r d ${_ip1} via ${_interf}
       if [ "$?" -ne 0 ]; then
         ip r d ${_ip1} via 10.55.10.3
       fi
       ip r a ${_ip1} via 10.55.10.3
       echo "New route added 10.55.10.3"
       _flg0=1
       _flg3=0
       _interf="10.55.10.3"
     fi
       elseif [ "${_interf}" = "10.55.10.3" ]; then
        if [ ${_flg3} -eq 0 ]; then
        ip r d ${_ip1} via ${_interf}
        if [ "$?" -ne 0 ]; then
          ip r d ${_ip1} via 10.55.10.10
        fi
          ip r a ${_ip1} via 10.55.10.10
        echo "New route added to 10.55.10.10"
         _flg3=1
         _flg0=0
         _interf="10.55.10.10"
    fi
  fi
 # avoid the dead loop
  let i=i+1
  if [ $i -eq 3 ]; then
   exit 1
  fi
 else
  sleep 2
 fi
 if [ $j -eq 10 ]; then
  j=0
  echo $j
 else
  echo $j
  let j=j+1
 fi
done
```

Figure 11: Snippet of Linux Script to monitor IP routes

Router#3 to monitor the status of its route to CS. If the router detects that CS is no longer reachable via Router#2, the program will automatically create the new route to CS*, which is now using the IP of CS. In our configuration, Router #3 constantly monitors eth0 (10.55.10.3) for connectivity with CS (10.55.12.241) by sending an ICMP message (ping request) to CS. After CS is no longer functioning, host unreachable messages with 100 percent packet loss are received by Router#3. The router then deletes the old route to CS and adds the new route to CS*. During this process, the client may lose connectivity for a short period (a few microseconds).

Implementation of the above was done in C/C++. The code sizes are similar to our previous designs [22-23] since only minimal changes were made. The implementation of the HTTP protocol and the necessary network protocols were based on a state transition diagram. The bare PC Web server runs on any Intel-based CPUs that are IA32 compatible. It does not use any hard disk, but uses the BIOS to boot the system. A USB is used to boot, load, and store a bare PC file system (raw files at this point). There was no need to change any hardware interface code during Web server modification to handle splitting or migration.

The software is placed on the USB and includes the boot program, startup menu, AO executable, and the persistent raw file system (used for resource files). The bootable USB containing this information is generated by a tool (designed and run on MS-Windows), which creates the bootable bare PC application for deployment.

6.3 Performance Measurements

6.3.1 Experimental Setup. The experimental setup in Figure 10 involved Dell Optiplex GX260 PCs with Intel Pentium 4, 2.8GHz Processor, 1GB RAM and Intel 1Gbps NIC card on the motherboard. The Ethernet switches were 1 Gbps 16-port Linksys. A Linux-based http_load [12] stress tool and a bare PC Web client were used for the experiments. Each http_load stress tool can run up to 1000 concurrent HTTP requests/sec. Each bare PC Web client can run up to 5700 HTTP requests/sec. A mixture of bare and Linux clients along with bare split servers were used to measure the performance of the split servers. Other popular browsers running on Windows and Linux (Internet Explorer and Firefox respectively) were also used to test the split servers for functionality.

6.3.2 Measurements. Our measurements focus on how many requests were pending when CS handed over, how long it took for the CS to migrate those requests to CS*, how long it took for those migrated requests to be processed at CS*, the relation between pending requests and the current CS load, and how much time was needed to migrate a server. As the split protocol servers are implemented on a bare PC, data collection was much easier.

Figure 12 shows a plot of the number of requests against the server load. The server load was measured in terms of the number of requests/sec for a 4KB resource file. As the load increases, there will be more requests waiting to be processed or delegated to DS. In our system, there are small number of pending requests at any given point in time as most of the requests were already delegated to DS as soon as their GET arrived. An inter-server packet with all the needed information is sent to DS for each delegated request. The experiments showed that about 1-138 requests were still pending in the server for a load variation of 1000 – 7000 requests/sec. These pending requests increase rapidly after

5000 requests/sec since it is getting saturated (higher CPU utilization).

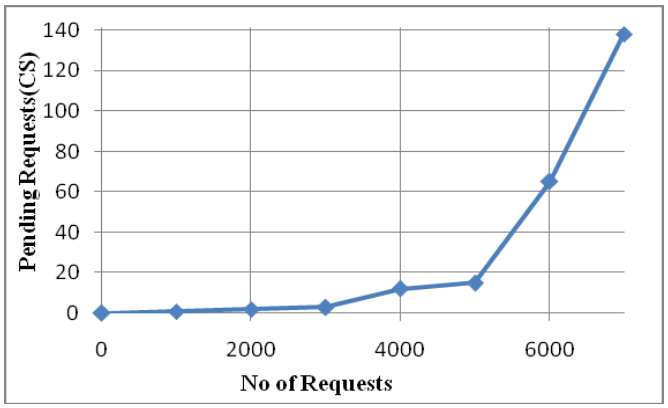


Figure 12: Pending requests at CS with 4KB resource file

In Figure 13, a graph showing the time spent in migrating pending requests from one CS to CS* is drawn. It takes between 1-13 milliseconds to migrate these pending requests from CS to CS*. Under heavy load conditions there is only a time lapse of 13 milliseconds to migrate all pending requests between the two CSs which are on different networks.

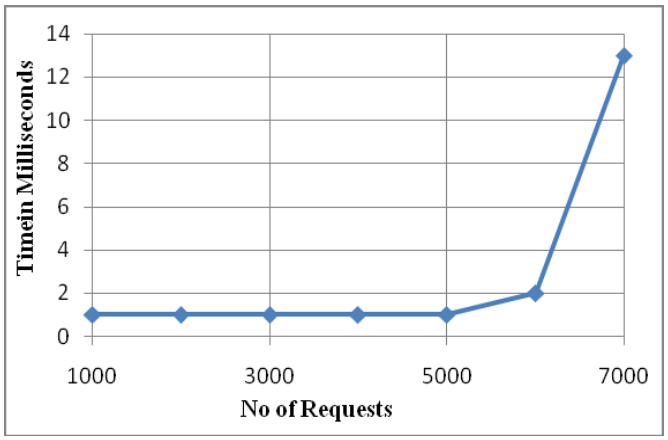


Figure 13: Time to migrate pending requests at CS with 4KB resource file

Figure 14 shows the number of pending requests for variable file sizes (4K to 128K) and a constant load of 500 requests/sec. Notice that the pending requests increase sharply up to 260 for a 128K file size. Larger file sizes have higher connection times (time to process the whole request) and thus more requests will be pending in the server as expected.

Figure 15 shows the actual time spent in migrating requests from CS to CS*. The times measured, range from 1 – 37 milliseconds for file sizes up to 128K. As the resource file size increases, more time is needed to migrate all pending requests as there are more requests in the server that are unprocessed. The migration time depends on the number of requests/sec (load) and the average resource file size.

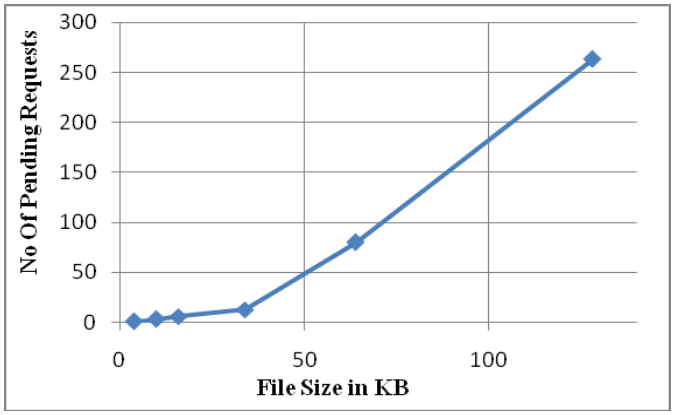


Figure 14: Pending requests at CS with 500 requests/sec load

Figure 15: Time to migrate pending requests at CS with 500 requests/sec load

To get further insight into migration delays, the pending requests in CS were simulated by artificially delaying request processing and collecting the migration time as shown in Figure 16. In this case, the load on the server is kept constant. Notice that the migration time is linear as expected.

Figure 17 illustrates the connection time and response time with and without migration for split protocol servers (CS and

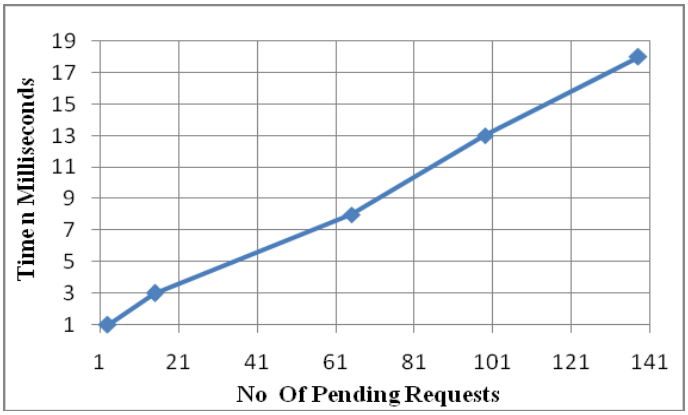


Figure 16: Time taken to migrate pending requests at CS

DS) with varying loads (requests/sec). CT and RT are the times with normal operation and CT-M and RT-M are the times with migration. The graph shows the degradation in connection and response time due to migration. It is seen that the increase in connection and response time is very small for loads of up to 3000 requests/sec. These times increase rapidly from this point due to the large load and the migration delay due to the increased overhead. A maximum load size of 5000 requests/sec was used in these experiments.

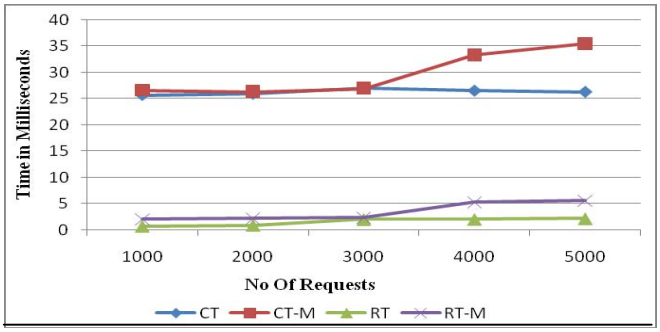


Figure 17: Connection time and response times (with and without migration)

Figure 18 shows the CPU utilization on the split protocol servers CS and DS. Notice that CS and DS reach over 90 percent CPU utilization for large loads (7000 requests/sec) with 4K resource file size. As shown in the graph, CS reaches saturation at 7000 requests/sec load. Further experiments are needed using migration with mini-clusters and splitting as in [23] to overcome the server saturation problem and to demonstrate the scalability of mini-clusters in the presence of migration.

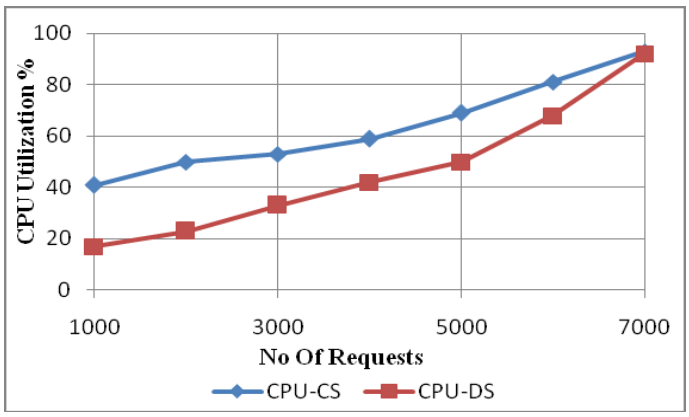


Figure 18: CPU utilization

Figure 19 illustrates the number of lost requests due to server migration. About 14-56 requests were lost for varying loads up to 7000 requests/sec. These requests are lost since it is not possible to process incoming requests during migration. The current server code was not enhanced to handle these requests; it is however possible to reduce lost requests by simply adding them into a pool of pending requests. In the current split design, pending requests only consist of requests with received GET commands from the client, so a different split request point is required to recover all lost requests. Once the migration process is complete, all lost requests will be reissued to CS*. Even with the current design, the above performance measurements demonstrate that the split protocol concept is well-suited for constructing reliable server clusters with inherent redundancy.

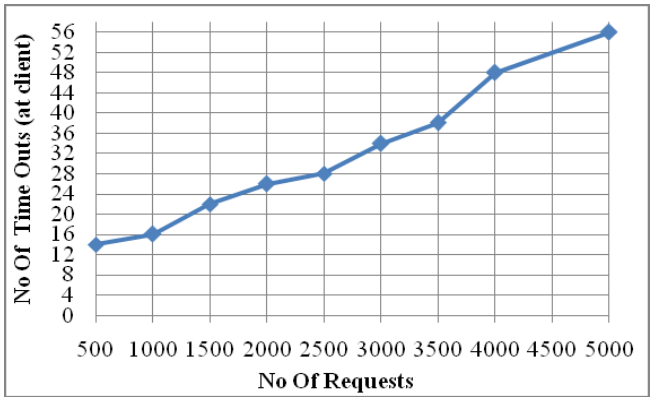


Figure 19: Lost requests at client

7 Significance of Split Protocol Concept

Splitting is a general approach that can be applied in principle to any application protocol that uses TCP (it can also be applied to protocols other than TCP to split the functionality of a protocol across machines or processors). In particular, splitting the HTTP protocol has many impacts in the area of load balancing. We discuss some of these impacts below.

- Split protocol configurations can be used to achieve better response and connection times, while providing scalable performance. Splitting also eliminates the need for (overhead/cost associated with) external load balancers such as a dispatcher or a special switch.
- Split protocol Configuration 2 (with one CS, one DS) and partial delegation achieves 25 percent more performance than two homogeneous servers working independently. This performance gain can be utilized to increase server capacity while reducing the number of servers needed in a cluster.
- Split server architectures could be used to distribute load based on file sizes, proximity to file locations, or security considerations.
- The results obtained (using specific machines and workloads) indicate that mini-cluster sizes are in the single digits. More research is needed to validate this hypothesis for other traffic loads. However, if we assume that mini-clusters should contain a very small number of nodes, they would be easier to maintain and manage (compared to larger clusters). Using mini-clusters, it is possible to build large clusters by simply increasing the number of mini-clusters.
- Splitting protocols is a new approach to design server

clusters for load balancing. We have demonstrated splitting and built mini-cluster configurations using bare PC servers. However, the general technique of splitting also applies to OS-based clusters provided additional OS overhead can be kept to a minimum (and that undue developer effort is not needed to tweak the kernel to implement splitting).

- When protocol splitting uses two servers (CS and DS), it dramatically simplifies the logic and code in each server (each server only handles part of the TCP and HTTP protocols unlike a conventional Web server that does both protocols completely). Thus, the servers are less complex and inherently have more reliability (i.e., are less likely to fail).
- Splitting can also be used to separate the "connection" and "data" parts of any protocol (for example, any connection-oriented protocol like TCP). In general, connection servers can simply perform connections and data servers can provide data. It can also be used to split the functionality of any application-layer protocol (or application) so that different parts of the processing needed by it are done on different machines or processors. Thus, a variety of servers or Web applications can be split in this manner. This approach will enable new ways of doing computing on the Web to be investigated.
- Split protocol servers used in migratory servers will provide reliability and simplicity which is inherent in the design.
- Migratory architectures designed on the basis of split protocol models can prevent expensive shadowing, backup and recovery techniques.

7.1 Security Issues

Protocol splitting and migration raise some security issues even though the servers are on the same LAN. When splitting is used, the DS sends packets back to the client using the source IP address of CS. This means that DS is allowed to spoof the IP address of CS in its outgoing packets. Likewise after migration, CS* will send and receive packets from clients using the IP address of CS. If CS* and CS are on different subnets, CS* is using an IP address whose prefix does not match the IP prefix of its subnet. Also, in our test LAN, we did not consider the impact on local clients and routers in subnets affected by giving CS's IP address to CS*. These local routers, for example, would need to be informed of the change so that they can route CS's packets to CS* (which is using CS's IP address in a different subnet). To minimize the security impact of splitting and migration, it is therefore best that CS, DS and CS* be located on the same subnet. If CS and CS* are located on the same subnet, the ARP broadcast by CS* is all that is needed to ensure that the network operates correctly after migration (it is not necessary to run the script for a route change). It is also necessary to authenticate inter server packets.

8 Conclusions

In this paper we discussed the novel concept of split protocols and their applications. Mini-cluster configurations and modified client server architectures were presented. Server migration and its impact on migration time and lost requests were measured. Full delegation and partial delegation concepts were used to achieve high performance. It is shown that split protocol servers inherently provide reliability, simplicity and scalability. Split protocol designs and migratory servers can be used to build high performance server clusters in the future.

References

[1] P. Appiah-Kubi, R. K. Karne and A. L. Wijesinha, "Design and Performance of a Webmail Server on Bare PC," *HPCC 2010*, pp. 521-526, 2010.

[2] L. A. Barroso, J. Dean, and U. Urs Hölzle, "Web Search for a Planet: The Google Cluster Architecture," *IEEE Micro 23*, 2:22-28, 2003.

[3] R. Baratto, S. Potter, G. Su, and J. Nieh, "MobiDesk: Mobile Virtual Desktop Computing, *Mobicom 2004*, Philadelphia, PA, 2004.

[4] G. Canfora, G. Di Santo, G. Venturi, E. Zimeo and M. V. Zito, "Migrating Web Application Sessions in Mobile Computing," *Proceedings of the 14th International Conference on the World Wide Web*, pp. 1166-1167, 2005.

[5] The Chromium Projects, available at http://www.chromium.org/, accessed: Feb. 3, 2014.

[6] G. Ciardo, A. Riska and E. Smirni, "EquiLoad: A Load Balancing Policy for Clustered Web Servers," *Performance Evaluation*, 46(2-3):101-124, 2001.

[7] A. Cohen, S. Rangarajan, and H. Slye, "On the Performance of TCP Splicing for URL-Aware Redirection," *Proceedings of USITS'99, The 2nd USENIX Symposium on Internet Technologies & Systems*, October 1999.

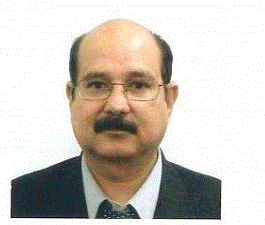
[8] D. Engler, *The Exokernel Operating System Architecture*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, October 1998.

[9] B. Ford, M. Hibler, J. Lepreau, R. McGrath, and P. Tullman, "Interface and Execution Models in the Fluke Kernel," *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, USENIX Technical Program, New Orleans, LA, pp. 101-115, February 1999.

[10] G. H. Ford, R. K. Karne, A. L. Wijesinha, and P. Appiah-Kubi, "The Performance of a Bare Machine Email Server," *21st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2009)*, IEEE/ACM Publications, So Paulo, SP, Brazil, pp. 143-150, 28-31 October 2009.

[11] L. He, R. K. Karne, and A. L. Wijesinha, "Design and Performance of a Bare PC Web Server," *International Journal of Computer and Applications*, Acta Press,

15:100-112, June 2008.
[12] http_load, available at http://www.acme.com/software/http_load, accessed: Feb. 3, 2014.
[13] Y. Jiao and W. Wang, "Design and Implementation of Load Balancing of a Distributed-System-Based Web Server," *3rd International Symposium on Electronic Commerce and Security (ISECS)*, pp. 337-342, July 2010.
[14] R. K. Karne, "Application-Oriented Object Architecture: A Revolutionary Approach," 6th International Conference, HPC Asia 2002, Center for Development of Advanced Computing, Bangalore, Karnataka, India, December 2002.
[15] R. K. Karne, K. V. Jaganathan, and T. Ahmed, "How to Run C++ Applications on a Bare PC," SNPD 05, 6th *ACIS International Conference*, IEEE, pp. 50-55, May 2005.
[16] R. K. Karne, K. V. Jaganathan, T. Ahmed, and N. Rosa. "DOSC: Dispersed Operating System Computing," OOPSLA 05, *20th Annual ACM Conference on Object Oriented Programming, Systems, Languages, and Applications*, Onward Track, ACM, San Diego, CA, pp. 55-61, October 2005.
[17] S. Lampoudi and D. M. Beazley. "SWILL: A Simple Embedded Web Server Library," FREENIX Track: 2002 USENIX Annual Technical Conference, Monterey, California, June 2002.
[18] J. Lange, K. Pedretti, T. Hudson, P. Dinda, Z. Cui, L. Xia, P. Bridges, A. Gocke, S. Jaconette, M. Levenhagen, and R. Brightwell, "Palacios and Kitten: New High Performance Operating Systems for Scalable Virtualized and Native Supercomputing," *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2010)*, pp. 1-12, April, 2010.
[19] D. S. Milojicic, F. Douglis, Y. Paindaveine, R. Wheeler and S. Zhou, "Process Migration," *ACM Computation Surveys*, 32(3):241-299, September 2000.
[20] The OS Kit Project, "School of Computing," University of Utah, Salt Lake, UT, http://www.cs.utah.edu/flux/oskit, June 2002.
[21] V. S. Pai, P. Druschel, and W. Zwaenepoel. "IO-Lite: A Unified I/O Buffering and Caching System," *ACM Transactions on Computer Systems*, ACM, 18(1):37-66, February 2000.
[22] B. Rawal, R. Karne, and A. L. Wijesinha, "Splitting HTTP Requests on Two Servers," The Third International Conference on Communication Systems and Networks: COMSNETS 2011, Bangalore, India, January 2011.
[23] B. Rawal, R. Karne, and A. L. Wijesinha, "Mini Web Server Clusters for HTTP Request Splitt," 13th International Conference on High Performance Computing and Communication, HPCC-2011, Banff, Canada, Sept. 2-4, 2011.
[24] B. S. Rawal, R. K. Karne, A. L. Wijesinha, S. Ramcharan, and S. Liang, "A Split Protocol Technique for Web Server Migration," International Workshop on Core Network Architecture and Protocols, (ICNA), 2012.
[25] B. S. Rawals, R. K. Karne, and A. L. Wijesinha, "Split Protocol Client/Server Architecture," IEEE Symposium on Computers and Communications, (ISCC), 2012.
[26] K. Sultan, D. Srinivasan, D. Iyer and L. Lftod, "Migratory TCP: Highly Available Internet Services using Connection Migration," *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pp. 17-26, July 2002.
[27] "Tiny OS," Tiny OS Open Technology Alliance, University of California, Berkeley, CA, I http://www.tinyos.net/, 2004.
[28] S. Vaidya and K. J. Chritensen, "A Single System Image Server Cluster using Duplicated MAC and IP Addresses," *Proceedings of the 26th Annual IEEE Conference on Local Computer Network (LCN'01)*, pp. 206-214, 2001.
[29] T. Venton, M. Miller, R. Kalla, and A. Blanchard, "A Linux-Based Tool for Hardware Bring Up, Linux Development, and Manufacturing," *IBM Systems Journal*, IBM, NY, 44(2):319-330, 2005.
[30] D. Wentzlaff and A. Agarwal, **"**Factored Operating Systems (FOS): The Case for a Scalable Operating System for Multicores," *ACM SIGOPS Operating Systems Review*, 43(2):76-85, April 2009.



Bharat Rawal is an Assistant Professor in the Department of Computer Science and Information Systems at Shaw University. He has a doctorate in Information Technology and an MBA from Towson University, and a BSc in Physics from South Gujarat University, India. His research interests are in split protocol systems and bare machine computing.

Ramesh K. Karne is a Professor in the Department of Computer and Information Sciences at Towson University. He obtained his Ph.D. in Computer Science from the George Mason University. Prior to that, he worked with IBM at many locations in hardware, software, and architecture development for mainframes. He also worked at the Institute of Systems Research at the University of Maryland, College Park as a research scientist. His research interests are in bare machine computing, networking, computer architecture and operating systems.

Alexander L. Wijesinha is a Professor in the Department of Computer and Information Sciences at Towson University. He holds a Ph.D. in Computer Science from the University of Maryland Baltimore County, and both a M.S. in Computer Science and a Ph.D. in Mathematics from the University of Florida. He received a B.S. in Mathematics from the University of Colombo, Sri Lanka. His research interests are in computer networks including wireless networks, VoIP, network protocol adaptation for bare machines, network performance, and network security.

Patrick Appiah-Kubi earned a doctorate in Information Technology from Towson University, an MS in Electronics and Computer Technology from Indiana State University, and a BS in Computer Science from Kwame Nkrumah University of Science and Technology, Ghana. He is an Assistant Professor in the Department of Electronics and Computer Engineering Technology at Indiana State University. His research interests are in bare machine computing systems, networks, and security.

Songjie Liang obtained his doctorate in Information Technology at Towson University. He is also an independent IT consultant, working for Lockheed Martin, Federal Government DOC, IRS, and USPS. His research interests include bare machine computing, file management systems, USB drivers, software engineering, database management systems, systems administration and Linux operating systems.

Using Relaxation to Fuse RFID and Vision for Object Tracking Outdoors

Rana H. Raza[*,†] and George C. Stockman[*]
Michigan State University, East Lansing, MI 48824 USA

Abstract

Fusion of Radio Frequency Identification (RFID) with Computer Vision (CV) can significantly improve performance in applications of autonomous vision and navigation, activity analysis, site monitoring, and especially in outdoor environments. RFID and CV provide both overlapping and unique information for deciding on object identity, location, and motion. We use relaxation to control the integration of information from CV, RFID, and naïve physics. Work site analysis must proceed even when information from one sensor or information source is unavailable at some time instances. Simulations show how fusion can greatly increase tracking performance while also reducing computational cost. Test cases show how fusion can solve some difficult tracking problems outdoors.

Key Words: Tracking, stereo, RFID, fusion, relaxation, site monitoring.

1 Introduction

We are interested in application problems in site monitoring, security, and activity analysis. Examples are tracking both baggage and people in airports; workers, materials, and machines in construction sites; or patients and care workers in medical care facilities. There are many other important applications. We believe we are the first to report experimental work on fusion of RFID and CV for object identification and tracking in an outdoor environment.

Figure 1 shows our outdoor test site: we are studying how well we can detect and track RFID tagged moving objects. The site is a courtyard roughly *40 Sq m* with several stone posts, sidewalks, and several large trees that limit both movement and observation via sensors. The surrounding building has many corners and windows. We surveyed landmarks using a combination of tape measure, laser range finder, and then stereo vision once enough calibration features were available. The same landmarks were available for calibration of a set of RFID readers as well.

1.1 Basic Functionality Required

The applications of interest require a system that provides some or all of the following basic functions.

a. **Detection** of the presence of objects of interest (persons, machines, materials, vehicles…).
b. **Identification** of the objects (by class or by a unique object instance).
c. Object **location** in workspace coordinates or by designated areas.
d. Object **track**, if the object is moving.
e. Important **object properties**, such as shape, color, weight, speed, ownership, supplier, etc.
f. A memory **representation** of space and time including location of objects, trajectories, and behaviors.
g. Application specific processes that manage objects and control their behavior (such as collision avoidance or creation).

Fusion of CV and RFID may provide the base functionality. Higher level problem-specific analysis applied on top of functionality (*a*) to (*e*) can create a dynamic inventory of a workspace, infer what agents or objects are doing (*function f*), actively manage interactions, define and summarize events, etc. (*function g*).

1.2 Some Advantages and Disadvantages of CV

Computer Vision has been very successful in controlled indoor environments, but challenged in uncontrolled outdoor environments. The CV literature contains thousands of reports on object detection and recognition, tracking, and motion analysis. Image sensors are passive, cheap, can be far-seeing, and can collect a good deal of information about a scene. Commodity cameras easily produce frame rates useable for most human motion analysis. Detections and relationships in a 2D image can often be mapped to the real 3D scene. Using multiple camera stereo, objects can be located in 3D(or, special active sensors can yield range/depth images).

Object identification via CV -- function (*b*) above -- is often difficult and is usually based on sensed features (*e*). Even accurate features may not precisely identify an object, e.g., who is that person or what year is that Chevy Cruz

[*] College of Engineering, Engineering Building, 428 S. Shaw Lane.
[†] E-mail: razarana@msu.edu; Phone 1 517 325-3260; ranahammadraza.com.

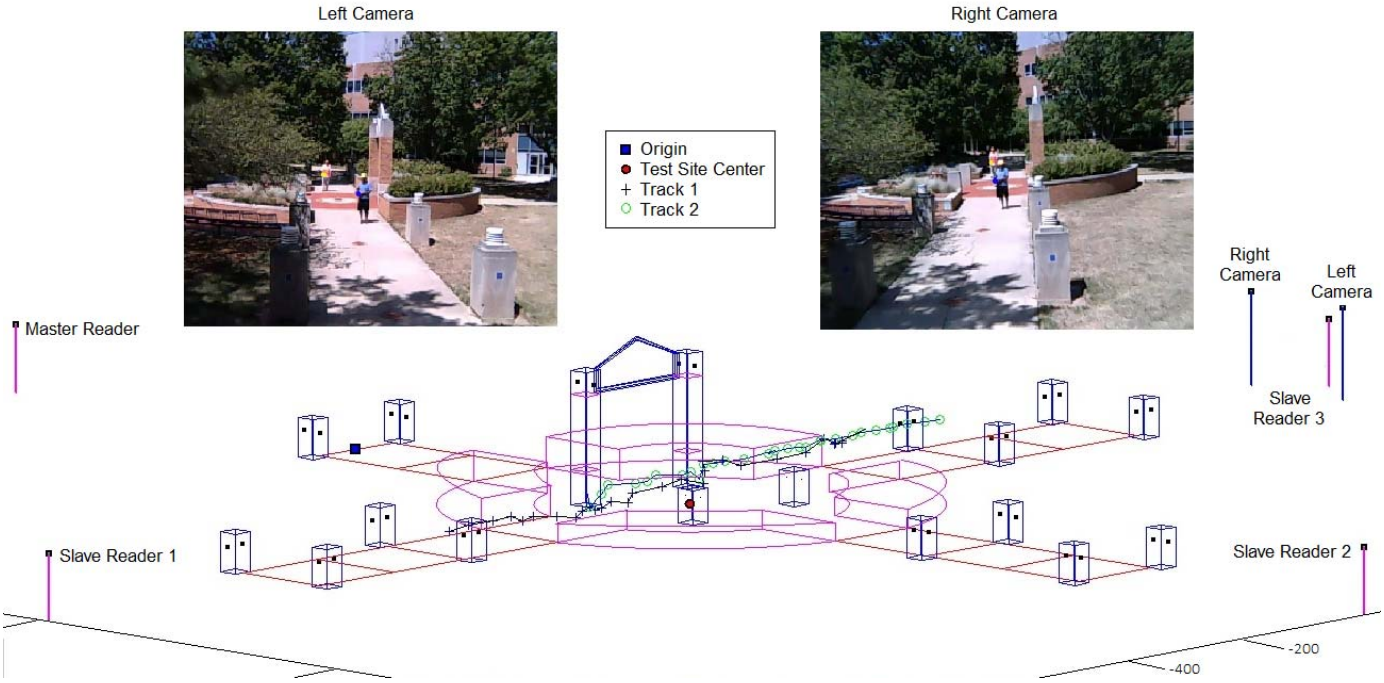


Figure 1: Outdoor workspace: about *40 Sq m* with trees and structures and a surrounding building

Identifying an object by features, if possible and reliable, can be computationally costly given the necessary signal processing and the variation of appearance over many possible 3D poses. So, while person identification using carefully imaged biometrics might yield identity accuracies of over 98 percent, more general object recognition via color camera image is far less accurate. Acquiring quality images under occlusion and variations in lighting also cause serious problems in CV applications. Uncontrolled outdoor environments might be dark, dusty, have rain or snow, and have both static and dynamic objects occluding the sensor view. Finally, CV may sometimes give too much information; for example, humans do not want to be imaged in private spaces, and may resent being watched in work places.

1.3 Some Advantages and Disadvantages of RFID

RFID can easily and reliably provide a unique object identification by transmitting a digital signal to a reader. With enough power, RFID can also transmit non visual features of an object, such as weight or ownership. RFID can operate in smoke, and darkness, thus it is widely used in sales and inventory systems and is replacing bar coding when cost permits. Object identification approaches 100 percent accuracy in commercial applications where objects are close to (presented to) a reader in a controlled environment. Objects with RFID tags can actually transmit their own physical description to an automated system or a security person. In robotics or material handling, the description might send a CAD model to a CV system to teach it how to recognize the object. RFID technology offers a wide variation in terms of cost, size, sensing distance, memory and processing power, and security [6]. With higher RFID frequency, higher data rate can be achieved. The higher data rate with appropriate anti-collision algorithm can enable a single reader to read a large population of tags. RFID can even be used to locate objects. In one method, a networked grid of short range RFID readers can be polled to find out which reader is sensing the nearby object. Alternatively, multiple long range readers can locate an object with an active tag by triangulation or trilateration [22]. The Real Time Location System [3] that we have used for our RFID sensing is discussed in Section 3. RFID codes cannot be sensed by humans and hence can yield less overt identity and might be tolerated in private human spaces.

RFID requires that an object be physically tagged, thus changing the object itself and requiring that the object be "cooperative". Although passive RFID tags can be tiny and do not require their own energy source, they are used for limited range and have limited memory. Highly functional RFID tags require an energy source for communication, memory, and processing. An RFID tag is a proxy for an actual object so it or its communication can be counterfeit, thus making an object appear to be what it is not. Simple examples of this would be one driver stealing an EZ-PASS device from another driver, a shopper moving a tag from a cheap article to an expensive one, or two airline passengers swapping their RFID boarding passes. Thus RFID tags in critical security applications need to use encryption and secure operating system principles. Fusion with CV can enhance security.

1.4 Outline of the Paper

Section 2 describes a few related prior applications and methods using fusion of RFID and CV. Section 3 describes

the model of the problem we approach and Section 4 highlights the relaxation scheme as its solution. Section 5 gives critical test scenarios illustrating the benefits of fusion to object tracking and further analysis. Section 6 is our concluding discussion.

2 Background

We have surveyed many different projects dealing with fused CV and RFID. All of the projects using fusion were applied in indoor controlled environments. Here we describe just a few significant projects that used RFID and CV and demonstrate the potential for many other applications.

2.1 Monitoring People in a Controlled Indoor Environment

Fusion of RFID and CV has been used in a day-care environment by S. Nakagawa, et al. [13]. Parents can view their child's activity via the Internet. RFID tags are placed on play objects and on children so that readers in the play space can identify and locate them. The system can then select appropriate cameras for good views of selected children and/or objects. Software alarms can be implemented for interaction between special pairs of objects and summarization of an entire day's activity of a child can be done. The parents can know who or what their child interacted with in a given day and can locate video segments of these interactions. There are other applications requiring similar functionality – elder care, studying how shoppers examine items for sale, or how visitors examine art in a museum. Teixeira, et al. [21] reported tracking patients in indoor elder care using cameras on the ceiling and cell phones. Statistical methods were used to correlate the image features with the locations reported by the cell phones system.

2.2 Robotics Applications

Passive stereo vision can locate detected objects in a 3D volume [19]. An RFID reader can be used to identify an object observed in some 2D image, thus aiding stereo; or, a network of RFID readers can provide coarse 3D location without cameras. RFID technology also enables smart objects to communicate information about themselves not available to optical sensors; for example, object weight, container content, etc. A tagged rigid object can even help provide an optical observer with a network downloaded CAD model of itself to be used for pose computation by the observer. This was done by Hontani, et al [7], who also used visual tags on objects as a starting point in matching an observed image of the object to a projection of the 3D CAD model. Chae, et al [2] using fusion of RFID and CV, proposed a global to fine localization algorithm for a mobile robot in an indoor environment. The problem of dynamic obstacle recognition in mobile autonomous platforms is addressed in [10] by using fused information. Limitations in mobile robots such as dead reckoning or wheel slippage are also addressable using fusion. Moreover, information can be written to an active tag perhaps recording object location and time, or perhaps what operation or measurement was done on the object.

2.3 Tracking Elephants in the Dallas Zoo

The commercially available Real Time Location System (RTLS) [3] can both identify and locate an active tag in a work or play space indoors or outdoors. Typically, five or more readers are distributed about the space, which can be as large as *600 Sq m*. An implementation at the Dallas Zoo provides displays to visitors that locate each elephant in a 2D map of their area [5]. Elephants wear an active tag on their ankle. The system can locate the tag (elephant) within *1* or *2 m* accuracy and update the location every *2 sec*. This is not an application of fusion with CV; however, cameras could easily be added and used as in the aforementioned day care application. It would be simple to place cameras about the elephant area and then add them to the area map so that they can be used to observe a particular elephant with a known location in the map. We use RTLS in our outdoor research on fusion and this RFID data is illustrated in Section 4. Evaluations of the used RTLS in both indoor and outdoor environments are available on the internet [9]. On a similar note an airport security system has been proposed and analyzed by Zekavat, et al. [24]. It is primarily based on RFID for location and tracking of passengers and staff.

2.4 E-Passports

Electronic passports have been designed that combine RFID tags with conventional printed information and a photograph [8]. The RFID tag can privately and quickly transfer information about the person (object) to a machine, thus streamlining information transfer and saving personnel time. Encrypting techniques can make forgery much more difficult. There must be a digital photo or fingerprint on the tag, which can be compared to the live person. Once a person has entered a secure area, the digital photo or fingerprint can be compared to live biometrics taken in the workspace, such as in doorways or stations, to verify the location or activity of the person. Early on, some reengineering of the E-passport had to be done to shield the RFID tag so that it could not be read by hackers in unofficial places when the passport was just slightly open.

2.5 Sensor and Cell Phone Networks

Wireless sensor networks are networks of compact, cost effective nodes that sense and communicate environmental conditions such as light and temperature etc. Many applications use sensory tags that are RFID tags, which incorporate sensory functionality in addition to identification and possibly localization. Sensor networks alone can be used to provide location information using relative location between sensor nodes [11]. In applications where location monitoring is required, sensor nodes are oriented with respect to a global coordinate system so as to provide geographically meaningful data.

Functionally cell phones are similar to active RFID tags and cellular towers are similar to RFID readers operating over

large distances. Commodity pricing brings impressive power to cell phones at moderate cost. The latest smart phones have many useful sensors onboard, such as cameras, accelerometers, gyros, compass, GPS receivers, proximity and light sensors. Also they have large memories and exchange arbitrary data across networks. These sensors can be used in a local setting to compute position and movement information, for example a high-accuracy acoustic based ranging system using mobiles [15]. It is reported in [1] that NTTDoCoMo has manufactured cell phones with built in RFID modules. The present and future needs of these systems being fused together will generate a single efficient device useful for the applications cited in this paper.

2.6 Using Naïve Physics in Tracking

Many studies have focused on processing video to compute features used to extract and track moving objects. For instance, authors in [25] have used a vision based image registration algorithm to compensate for camera motion and then consecutive frames are transformed to the same coordinate system to solve the feature point correspondence problem. Zhou, et al. [26] have provided a blob tracking and classifying method in an outdoor environment by establishing correspondence between the object in view and the matched templates. A method of background subtraction and shadow detection in videos is provided in [4]. Meier, et al. [12] have reported an algorithm that can automatically extract moving objects from an image sequence. A survey of passive monocular methods is given by Veenman, et al. [23]. Consistency of color, texture, shape, and motion can be used to track an object region across multiple video frames. Variable lighting, variable 2D projections of a 3D object, and occlusion of one object by another present difficulties. Applications that need to recognize what the objects are face additional uncertainty and complexity. For example, an autonomous vehicle needs to identify obstacles in its path using their image extent and their motion or apparent motion [14]. Passive tracking using only cameras remains an important area of continued research.

3 Problem Representation

Our problem is to compute in real time the trajectories of N objects moving within a known 3D workspace. Diverse sensor observations are combined into object locations, and possible identities, at discrete time steps, which must be aggregated into N object trajectories. Without loss of generality, we may ground our discussion using the Site Safety System (SSS), where we want to track workers, materials, and machines in a work site. We abstract the information structures in order to support a system with diverse information sources and constraints and processes that may not have knowledge of each other.

3.1 Tokens Code Observations from Images and RFID

Sensor observations, and combinations of them, produce ***tokens*** $\boldsymbol{\tau}=<x, y, z, t, v, L>$, each coding that an object with identity, or name, L and feature vector v is at location (x,y,z) at time t. Some tokens will have incomplete information: for example, Identity L may be absent from camera observations and object features may be absent from RFID observations. 3D coordinates may be absent for an observation from a single camera image or single RFID reader. Two or more of these tokens can be combined in the processing to get 3D coordinates.

3.2 Object Tracks {<x, y, z, t, v, L>}

The site safety system SSS must detect, identify, and locate objects in a few video frames k. SSS may know $L = f(<x, y, z, v, t>)$ from sensor subsystems that use RFID or visual features. SSS can also use "tracking" to determine the label $L = f(\{<x, y, z, v, t-k>\})$ based on prior tokens or perhaps even forward tokens $\{<x, y, z, v, t+k>\}$. If object identity L is known, other object features $w = f(L)$ may be available from an RFID tag, such as object mass or CAD model. Finally, we note that if sensors supply object speed or acceleration, as cell phones may, we consider these as components of v along with color, texture, elongation, etc. of its image.

An **object track** is k or more tokens in time sequence with consistent object identity and features that also satisfy constraints for motion in space. Tracking is an important concern of this paper, and is a low level of motion understanding that uses naïve physics to aggregate observations over time. Heuristics from naïve physics enable aggregation of individual tokens into a sequence or track, one for each moving object. As objects move through the workspace, they may be occluded at any instant from either cameras or RFID readers so there may not be multiple tokens to fuse. Smoothness constraints, or motion applied over multiple time steps can be used to interpolate.

As we will see, it is not possible to assign unique object identities to every token at every time instance. Consider, for example, the popular shell game where a bean is placed under one of three shells that look alike [20]. When the shells are shuffled quickly in space, most people cannot track the shell containing the bean. If the shells are of distinct colors, then the problem of picking the final shell is easy. If the shells are identical in appearance, but the bean is an RFID tag, RFID readers are unlikely to be able to distinguish the tagged shell in space when the shells are close to each other. Consider three workers with hard hats each with a tag and close together; if the hats are the same color, RTLS cannot distinguish them; if we know which colors contain which tags and the hats are of different colors, the system can solve the matching problem and locate each hat within the CV distance error. In order to model ambiguity, we will have to allow multiple labels L in the tokens of an object track: these labels record the ambiguity of idenity at this point in time and space.

3.3 Obtaining 3D Object Location (x, y, z)

All sensors are calibrated to the same 3D workspace. One fundamental sensing concept is that a sensor observes an

object along a ray in 3D space. The object can be located by intersecting two (or more) rays (or error cones) as done by using the standard stereo solution [19, Ch 13]. Possibly, two RFID readers, or one RFID reader and one camera can be combined this way as well. The underlying geometry is angle-side-angle, where the side is the known 3D baseline between the two sensors. A second fundamental sensing concept is where the sensor observes an object at some distance *d*. If the object transmission is observed by four such sensors, it can be located by trilateration, intersection of four spheres with radii equal to the sensed distances. An object can also be located by intersection of the ray/cone determined by an image observation and the spherical shell determined by distance *d* sensed by a single RFID reader. The commercial RTLS system encapsulates multiple RFID readers and yields a token with unique object identity and *(x, y)* coordinates on the ground plane of the workspace.

3.4 Fusion

We define fusion as the combination of different sensor tokens to obtain a token containing information from the different sensors or with new information computed from the tokens from the different sensors. Most importantly, RFID and CV tokens will be fused to combine object identity with object features and to provide or to refine object location.

3.5 Naïve Physics

Naïve, or common sense, physics provides many constraints about the world. These constraints have diverse forms and can be used in the filtering processes of relaxation. An object *n* must be at one and only one place at time *t* and location <*x*,*y*,*z*> can accommodate at most one object at time *t*. Object *n* is likely to have consistent form and visual features and observations of object *n* must be consistent with its identity. The motion of object *n* is likely to have smooth direction and velocity. Such constraints extend those used by Sethi and Jain [18] and Veenman, et al. [23]. Unfortunately, none are hard constraints! For instance, it may actually be that two objects have the same coordinates – when a driver enters a vehicle, for example.

3.6 Sensor Error and Synchronization Problems

In an outdoor environment we evaluated positional accuracy and reliability for the RFID based Real Time Location System (RTLS) and the stereo computation obtained using commodity cameras [17]. The calibration of stereo system and RTLS system were done on the same test site. An adequate number of distant points (in the background) and nearby points (in the foreground) were acquired to serve as calibration markers for stereo computation. The stereo infrastructure provided RMS positional accuracy of *19.3 cm* for *x,y* and *z* directions. The reported location accuracy for the RTLS system for static tags is *~1.5 m* and for dynamic tags is *2 m ~ 2.6 m*. RMS error does not include the occasional outliers that are possible from incorrect stereo correspondence or multiple path effects in RFID.

One significant practical problem for fusion is the different sampling rates of the sensors or the extended time needed to smooth data or to make decisions about the motion of an object. Due to time division multiplexing, our RFID system provides data on all objects every *2 sec*, while our stereo implementation could produce *10* updates per second for a few objects. In our experiments we typically force a common sampling time for RFID and CV and look back two time samples to estimate motion. The uncertainty of location for RFID is much larger than for CV for static objects and even larger for moving objects due to under-sampling. Interpolation using CV locations can be used with sparse RFID samples with reliable identity. Finally, it is possible that an object is invisible at some time steps to either or both CV and RFID due to occlusion.

4 Relaxation Labeling Scheme

We use discrete relaxation to create the tracks of the *N* objects and to update the time tokens comprising each track. Using relaxation, different sensors and sources of information can be turned on or off for experimentation or for practical reasons at a site. Fusion processes operate on a blackboard containing the set of tokens. When an observation is made, its initial label set is the set of all possible *N* known objects. Filtering processes are then applied to eliminate labels inconsistent with constraints. Sensing continues over the *T* time steps and naïve physics processes aggregate object consistent tracks.

For clarity, suppose that *N* objects are detected at time *t=1* and that we arbitrarily label these objects *1,…,N*. At time *t=2*, we have another *N* observation and we want to label each of those with the labels from time *t=1*. A label possible for a token at time *t=2* will be consistent in color, motion, and RFID identity with the tokens at time *t=1*. Initially, a new observation may detect any of the known objects, so all labels *L* are possible. A totally new object entering the site could be given a new unknown label. Most of these labels are filtered out quickly by failing constraints. For example, suppose *5* orange hard hats are detected at *t=1* and these have initial labels *3,4,6,8,9*. For any token for time *t=2* that is not orange, labels *3,4,6,8,9* will be deleted from its possible label set. Filtering can be done by space as well as by color. If any token at time *t=2* is unreasonably far from a token *m* at time *t=1*, then label *m* should be deleted from its label set.

4.1 Sensor Processes

A CV sensor process takes a video frame, segments it, and creates a token for tracking. Image features and two points on the imaging ray are stored in the feature vector *v*. Object *L* is initially unknown. An RFID reading produces a similar token, except that an object label *L* is known in almost all cases.

4.2 Combination Processes

Fusion processes take the sensor tokens and possibly merge

information using ray intersection, ray-surface intersection, etc., whichever applies, and outputs a token with refined 3D location or label information. Filtering processes eliminate unlikely token labels by comparing tokens and by looking at feature vectors over time. (Our current software implementation of relaxation inputs combined tokens that have been pre-computed from stereo correspondence. Similarly, RFID tokens have 3D information from the encapsulated RTLS system.)

4.3 Tracking Process

Naïve physics constraints are used to filter out highly unlikely labels for objects at time *t* based on the recent history of objects continuing from the *k* previous time steps. Our current results have used the current and *2* previous time steps.

4.4 Relaxation Labeling Algorithm

Output: Object Labels L_k and 3D location $XYZ_{Refined} \in R^3$
Input: Object Labels L_{k-2} and L_{k-1} with color, RFID and XYZ_{RFID} and $XYZ_{Stereo} \in R^3$

FOR $t = k : K_{frames}$

- Obtain color information if any for XYZ_{Stereo} observations from 2D histogram matching
- Sort colors into groups */* How many colored hats and balls and which colors*/*

Detect

- *n* number of XYZ_{Stereo} observations detected */* Motion detection and color detection*/*
- *m* number of XYZ_{RFID} observations detected */* Active RFID*/*
- Generate empty label matrices for *p* observations */* p = max(n,m)*/*
- Assign *p* labels to all *p* observations and proceed to next pass

Identify */* Binary relationship criteria*/*

- Identify XYZ_{Stereo} observations based on color information
- Identify XYZ_{RFID} observations based on identity
- Correlate identity information

IF Only one color group /**All* XYZ_{Stereo} *observations have same color**/
No label elimination and proceed to next pass
ELSEIF Different color groups */*Some* XYZ_{Stereo} *observations have different color*/*
Eliminate labels from *p* label matrices based on respective color groups and proceed to next pass
END IF

Locate */* Binary relationship criteria*/*

- Set stereo and RFID location threshold values */*Thresholds are defined based on sensor location accuracy and object speed*/*
- Locate XYZ_{Stereo} observations
- Locate XYZ_{RFID} observations with identity and location
- Correlate location information

Smooth

- Calculate direction of flow/velocity for every XYZ_{Stereo} observation at $t = k$ relative to *k-2* and *k-1*
  / z dimension gives valuable information here */*
- Correlate with RFID label/s identity and location information from XYZ_{RFID}

IF No difference in flow detected
Labels kept
ELSEIF Difference in flow detected */* Only compatible labels remaining.*/*
Eliminate unlikely labels
END IF

Compatible label/s obtained */*All possible labels for specific object*/*

- Compatible label/s provided to optimization process to obtain $XYZ_{Refined}$

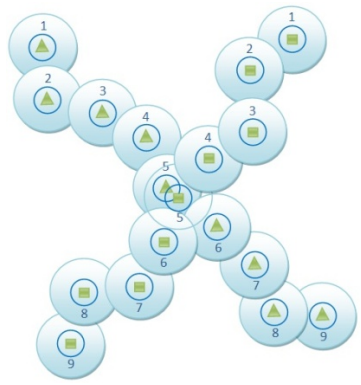
END FOR

5 Test Cases and Analysis

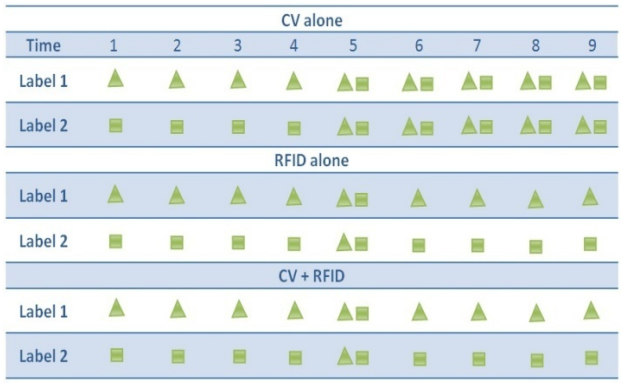
We illustrate in this section how CV and RFID supplement each other in critical test cases. For clarity we first assume that for case I and II, both CV and RFID feeds are continuously available and the objects are not occluded by each other or the background and are moving with approximately the same velocities. Actual observations from our outdoor test site are used.

5.1 Test Cases to Explain Fusion

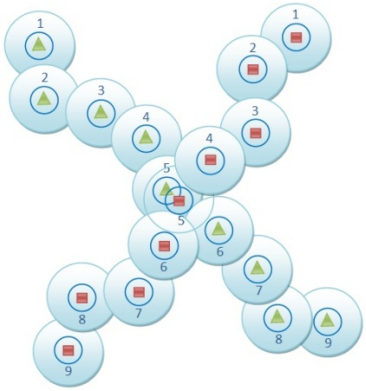
For case I, consider two objects represented as ▲and ■ with 3D data at each instance over 9 time frames. For better visualization the tracks are displayed in the *xy-plane* in Figure 2a. The objects are converging from north to south towards each other, and intersect at time frame 5 and thereafter follow their direction of motion without any change. Even if the objects were moving in a straight line, the points would appear



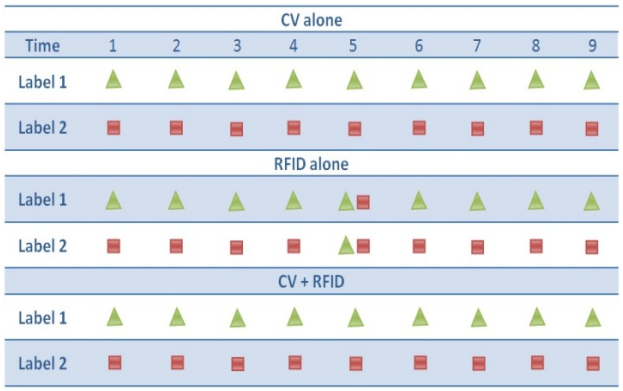
(a)



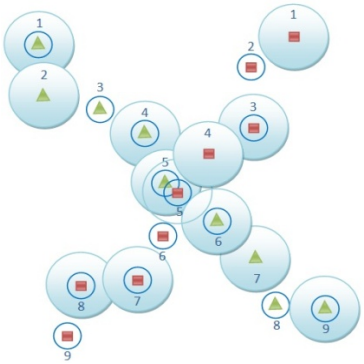
(b)



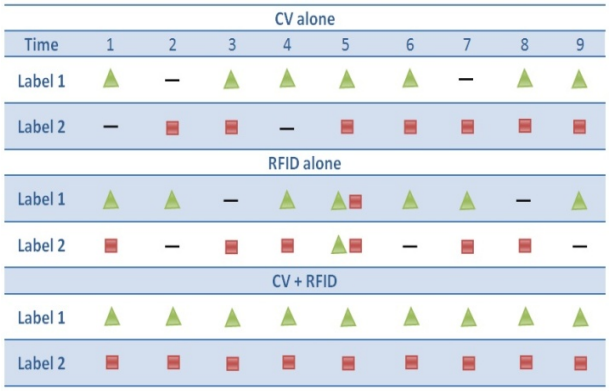
(c)



(d)



(e)



(f)

Figure 2: CV and RFID supplementing each other: (a) Same colored object tracks with label assignments in (b). (c) Different colored object tracks with label assignments in (d). (e) Considering visual occlusion and intermittent RFID, different colored object tracks with label assignment in (f)

to be scattered along the true path due to propagating location errors and distortions in a 3D space. CV and RFID location accuracies are shown with circles. The inner circle around every point shows the localization error of CV and the outer circle represents that of RFID. Figures 2a and 2b shows case I where both the objects are of the same color. Figure 2a shows the object tracks and Figure 2b represents label assignments. The CV system can correctly assign labels to ▲ as label *1* and ■ as label *2* up until time frame $t = 4$. Thereafter, there is a probability of no identity assignment, which based on relaxation labeling means no wrong label elimination and is represented here using both labels for both points. On the other hand, RFID provides correct label assignments other than at $t = 5$ due to fully overlapping localization error of point ▲ and ■. In this case RFID helps CV to generate correct object tracks. However, no label elimination in the intersection area is possible. The only contribution of CV is that it refines location.

Figures 2c and 2d shows case II where objects are of different color. Due to no occlusion CV will be able to provide correct label assignments. However, RFID will have no label assignments at $t = 5$. Fusing both feeds, CV supplements RFID here and the label assignment at $t = 5$ is obtained. For both cases CV support can also be clearly appreciated when RFID location error is maximum (i.e, on outer circle boundary) for two points having overlapping localization error in consecutive frames.

Figures 2e and 2f showcase III where some of the objects are occluded by the background and the RFID feed is intermittent. This is represented as missing vision and/or RFID location accuracy circles. The dash symbol shows non-availability of observation for that time instance. Comparing Figure 2e and 2f it is obvious that CV and RFID supplement each other at missing spots and fusion of these generates correct label assignments.

To realize how the dynamics of relaxation labeling can fuse information we describe here some related critical test cases.

Figure 3 shows case IV with simpler dynamics where two objects having the same color features moving from west to east first converge, move side by side for some time, and then diverge. The number of possible compatible labels after fusion is shown below each block of time frames.

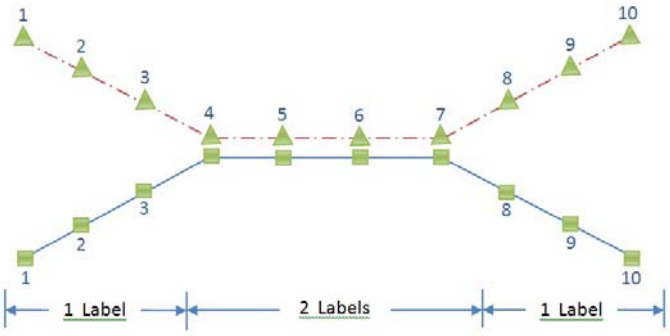


Figure 3: Correct object tracks with possible compatible labels at each block of time frames

Consider case V that is reconfigured from the scenario given in [17]. Two persons ▲,♦ (wearing distinctive clothing) carrying two balls ●,■ move towards each other and meet at the center of the test area. They then exchange the balls and backtrack to their starting positions. Both persons and both balls are tagged. To test algorithm robustness and under increased complexity we consider that the color of the balls and the persons head gear is the same. For better representation the observed 3D points shown in Figures 4a and 4b are the simulated version of the scenario over consecutive time frames.

Figure 4a represents correct trajectories of the persons and the balls. If we assume that there is no occlusion and only the stereo feed is continuously available then Figure 4b shows incorrect trajectories of the persons calculated by the stereo feed alone.

It is assumed that we have prior information about the feature set and 3D location of the object labels at time frame $t = 1$ and *2*. For subsequent time frames we correlate CV and RFID information and apply constraints in *detect*, *identify*, *locate* and *smooth* passes. Constraint based label elimination by these filtering passes update the label matrix for every observed point at every time instance. Once all impossible labels are removed and no further elimination is possible then the remaining labels are considered as compatible label/s. The labels are then passed on to the post-processing optimization process for updating fused token feature vector v and the refined location XYZ and, where required, determining a possible unique label in the compatible labels set. The labels acquired are then assigned to the observed points respectively. Figure 4c demonstrates a typical label matrix on the left that shows all four passes with the remaining compatible labels at the end. The RFID location information on the right is shown with each label matrix to provide evidence of objects presence.

For the observations in Figure 4a a step-by-step explanation on how the label matrices are updated is provided in Figure 5. For time frame $t = 3$ and *4* in each label matrix the objects are detected in the *detect* pass based on motion, color and identity and subsequently all the possible labels are assigned to all the observed object points. In the *identify* pass the system identifies objects based on color groups and identity from RFID. Since the color information for the observed points is the same, it is considered that no label is eliminated at this pass via color histogram based similarity. In the *locate* pass the labels are eliminated based on near neighbors where thresholding is done using sensor location accuracy and object speed. This helps identify ▲,● and ♦,■ as consistent label pairs. The two inconsistent labels are then eliminated from the respective label matrices. The *smooth* pass correlates labels with RFID and deletes one further label with unlikely motion according to local (*3 point* or *2 point*) smoothness and object height constraints. This would leave more global tracking to post processing after all the relaxation is completed. Note that at $t = 5$ the process of label elimination is complex due to the overlapping location errors of stereo and RFID making label elimination impossible in the *detect*, *identify* and *locate* passes. During the *smooth* pass, RFID provides no label elimination

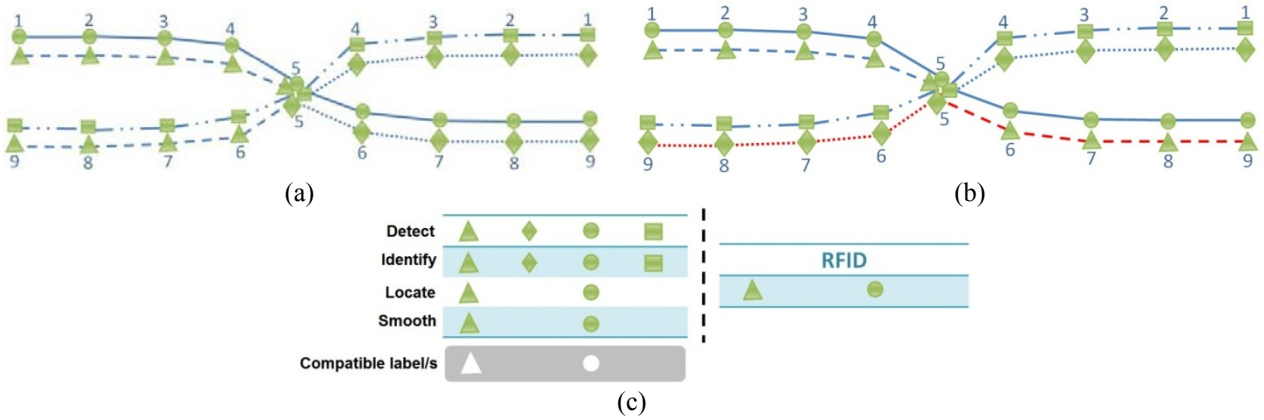


Figure 4: (a) Correct trajectories of persons and balls. (b) Correct balls and incorrect persons trajectories. (c) Left matrix - General pattern of four relaxation constraint passes and final compatible labels. Right matrix - RFID location information

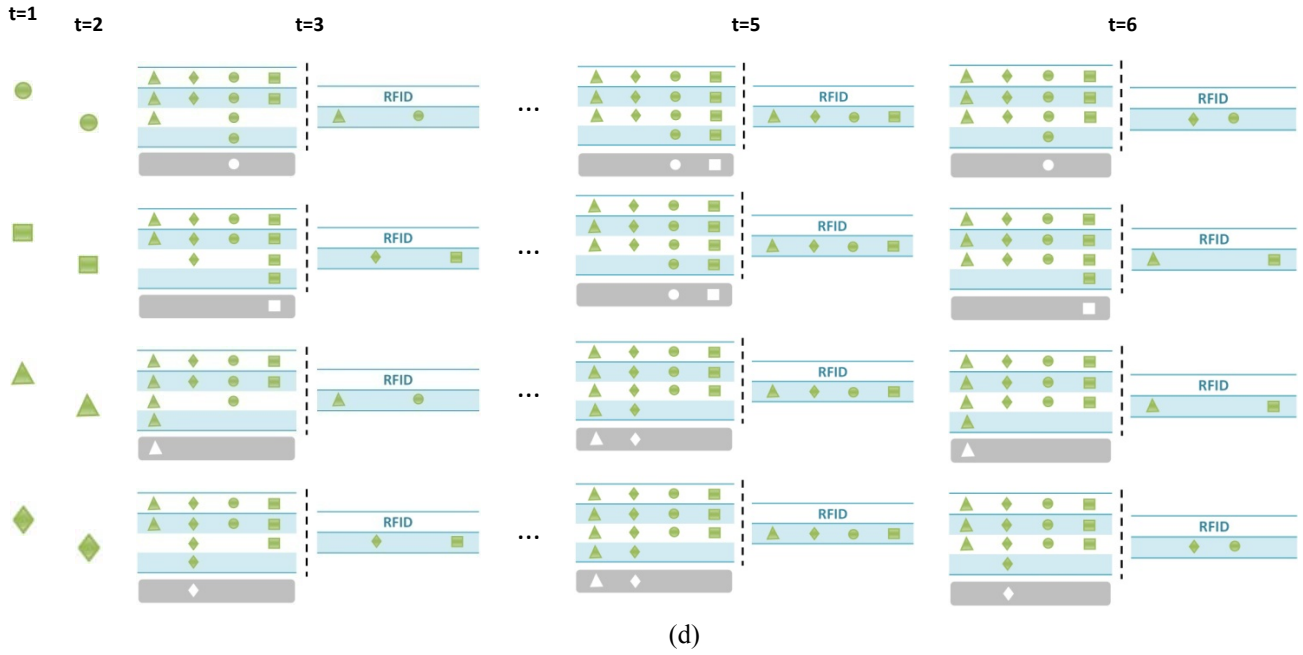


Figure 5: Label matrix updating steps for same colored objects at each time frame for Figure 4a tracks

information showing all four labels ▲,♦,●,■ as valid, however, the system identifies ●,■ and ▲,♦ as possible label set pairs based on object height and velocity constraint and subsequently outputs two compatible labels.

The compatible labels are then fed to the post optimization process to identify the optimal label for each observation. Note that the system keeps one extra label as part of the possible compatible label set. This explains a tradeoff between increased post processing computation for keeping a wrong label and the cost of eliminating a correct label. Since the objects have the same color and are assumed to be moving with the same velocity at $t = 6$ the color and near neighbor constraints will not provide information for label elimination. In the *smooth* pass based on height and direction of flow relative to the previous velocity vector direction, CV identifies ▲,●, and ♦,■ as compatible label sets for respective observations. These two label pairs for each observation represent correct trajectories of the balls, but incorrect trajectories of the persons as shown in Figure 4. However, RFID provides ♦,● and ▲,■ as possible label pairs. Correlating this information helps obtain one correct compatible label for each observation.

5.2 Object Color Variations

We collected various samples and analyzed HSV color space consistency for the blue and yellow balls in different weather (winter and summer) and illumination (sun and shade) conditions as shown in Figure 6.

The results shown in Figure 7 show the color consistency for

Figure 6: Different weather and illumination conditions

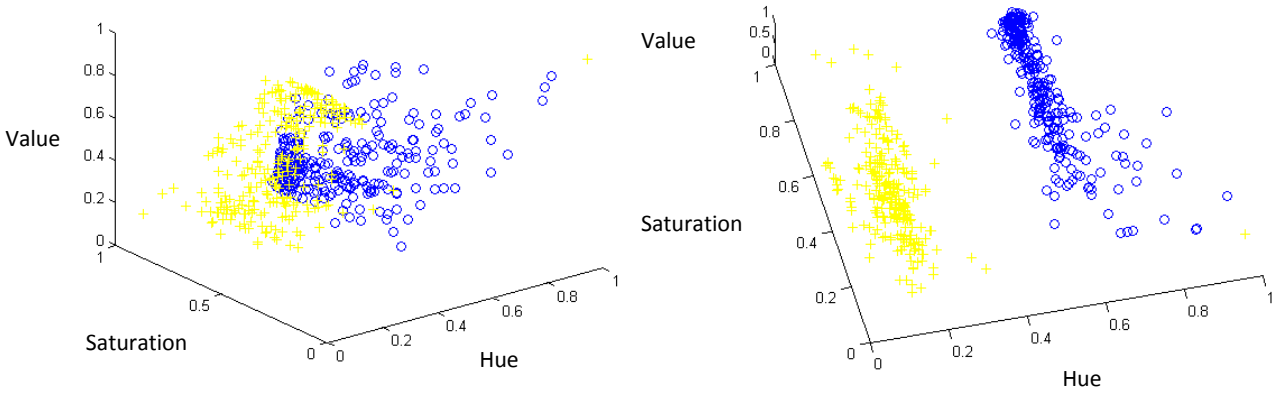


Figure 7: Analyzing blue (O) and yellow (+) ball color consistency in HSV color space under different weather and illumination conditions

blue and yellow balls for reliable color clustering. Yellow ball HSV value is represented by + and the blue ball HSV value is represented by O. The color clusters are clearly separated along the hue axis, which proves usefulness of CV to help distinguish objects based on color in an outdoor environment.

The irregular outdoor illumination variations and abrupt changes of brightness is evident in Figure 6. If color is to be used by CV to help tag and distinguish objects, then the objects must be for the most part distinguishable in the video images. In many cases workers will be wearing hard hats or

vests of special coloring. The SSS should be able to take advantage of these distinctive colors by exploiting color consistency for reliable color clustering.

The experiments reported in this paper did not use automatic color similarity computations to distinguish the class of object color: instead, a symbolic color was assigned to the token.

5.3 Simulations of Object Tracking

Prior to collecting real outdoor data, we performed many simulations in order to assess how effective labels could be in tracking under smoothness constraints – using observations of location but not color. We created many ground truth object paths using real stereo observations made in our calibration track volume. A brightly colored ball was waved within a *69x81x61 cm* track volume and the stereo system computed the path of the ball in 3D. This was repeated ten times so that we had ten paths within the same workspace [16].

Resulting ground truth paths are shown in Figure 8. We could then take subsets of these paths for simulations. *N* observations over the time steps *1...T* were selected and presented to our tracking algorithm to see what tracks would be aggregated using the naïve physics constraints. Smoothness of trajectories requires a burst of time frames to reliably compute track smoothness, curvature, and acceleration.

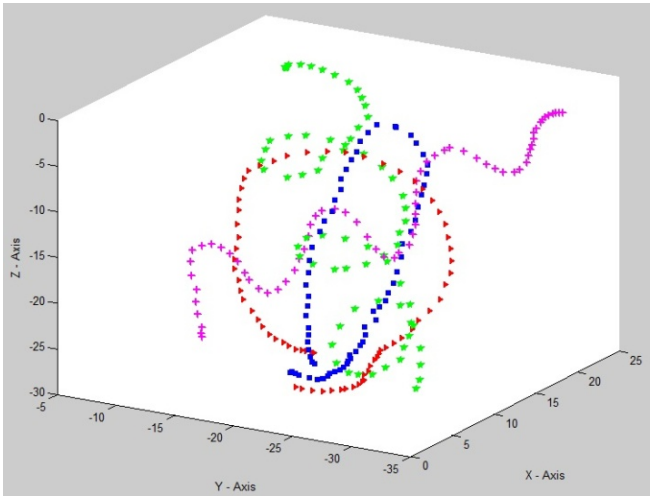


Figure 8: Ground truth trajectories generated using real stereo rig in *69x81x61cm* track volume

If we consider *n* objects and a burst of *m* time frames, then the number of possible paths will be $(n)^m$. Assume that *T* is divisible by *m*. If there is no identity information available then the number of combinations for *T* time frames will be $(T/m)x(n)^m$. Depending upon the probability *P* for an observation identity being available for the burst, the combination volume is reduced accordingly. It is considered that the identity when present is available for the whole burst. For example with *n = 3*, *m = 4* and *T = 60* the total combination volume will be *1,215* possibilities. As shown in Figure 9, with *P = 0.267* the combinations volume is reduced upto *435*.

Simulations were conducted using *N=5, 6* and *10* object tracks and *T=60* time steps. Using probability *P*, the ground truth identity was provided in the token. Figure 10 shows results for reduction in combination volume with increase in probability *P* of object identity in the token. Computation time is also shown at marked places to realize the reduction in volume. With respect to our outdoor experiments, the probability *P* represents the time percentage for which the RFID feed for a tag was available. The algorithm was run with frame burst length *m=4*. Identity of the bursts was assumed to be randomly available. Figure 10 shows that while tracking *10* objects the combination volume can be decreased up to *99.9 percent* with the partial identity feed thereby reducing computation time.

The effect of having some identity in the tokens increases as the number of object tracks *N* increases. This data shows the difficulty faced by tracking algorithms that only use motion of image points to aggregate object tracks. Without any object identity, quantifying motion over several time steps leads to too many possible tracks. Although color, shape and texture features can be used by a passive CV system, the reliability of unique labels from RFID can yield correct tracks with far less computation. These simulations motivated us to implement an actual Site Safety System using fusion of CV and RFID.

6 Concluding Discussions

We have argued that the fusion of CV and RFID can produce more accurate object tracking and do so using more efficient computation. The basic reason is that RFID can provide highly reliable unique object identification, although with coarse object location, while CV can provide more accurate object location along with confirming visual features. We

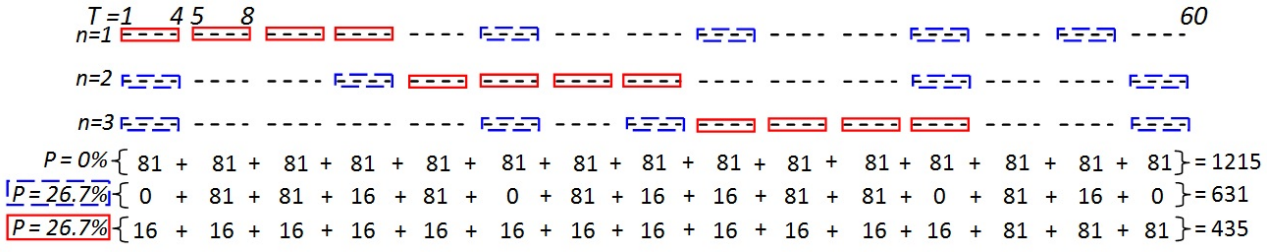


Figure 9: Reduction in combinations with probability of random identity information availability

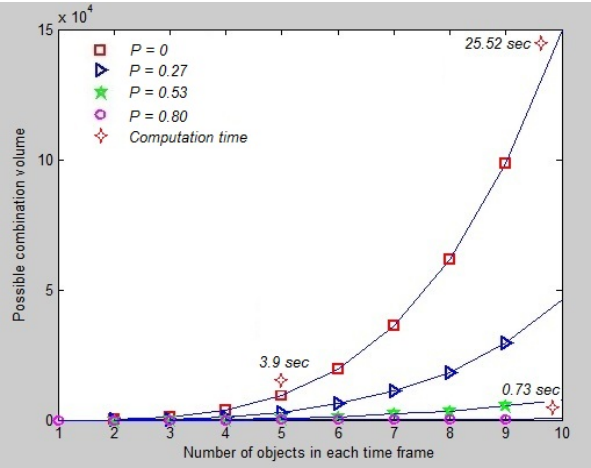


Figure 10: Possible combination volume with *n* objects and probability *p* of object identity in bursts of 4 tokens

have performed fusion experiments with tagged moving objects in a complex outdoor environment and the results support our predictions. For RFID we have used a commercially available Real Time Location System [3] and we developed our own stereo system with a laptop, MATLAB, and two commodity color cameras. Our total hardware cost was only about *US$5500*, for both the RTLS and only one stereo pair of cameras. High level performance would require more cameras and more RFID readers in the workspace than we have used. One significant problem in fusing the RFID and CV feeds is the difference in sensing frequency. Commodity cameras are designed to represent human motion well and produce upwards of *10* video images per second, whereas our RTLS system produced tokens for all tags at *2 sec* intervals. Engineering faster RFID updates will likely reduce the number of objects that can be sensed; however, this should be a favorable tradeoff in a construction site. It may also be good design to have a hierarchy of RFID sensing with a slow system for asset/material inventory and a fast system for critical objects such as workers and moving machinery.

We established that the RMS location accuracy of our stereo system is *19.3 cm* in *x, y,* and *z* for trajectories upto *24.4 m* from the cameras in a workspace that is *40x40 m*. The location accuracy for RFID was about *1.5m* in *x* and *y* ground coordinates for static objects, but about *2 m to ~ 2.6 m* for moving objects, which we attribute to the location update frequency. These results are consistent with previously published tests [9]. A commercial system should cover the workspace with a network of cameras. We have demonstrated cases where fusion disambiguates object tracks and we have also given cases where disambiguation is impossible, as in the well known shell game. We demonstrated how uncooperative objects can cheat the system. However, in general fusion of RFID and CV is better than using only one mode alone and, where costs are justified, will produce systems that are better than those using only one modality. Moreover, an automatic system detects the ambiguities and can cue the attention of higher level processes or longer lived processes, including the attention of human security personnel.

Simulations of tracking over many ground truth paths demonstrates how knowledge of unique object identity for some time instances can significantly improve correct tracking as well as reduce computation time in producing the tracks. Thus, many more objects can be tracked in practice if fused sensing is available compared to tracking by CV alone. A fast tracking implementation would be active – it could plan more efficient work, warn of possible collisions, or detect illegal operations. Finally, it is clear that the global workspace view we have used is too imprecise for detailed object interactions, such as cooperation compared to collision, or handing off carried objects. Object born touch or looming sensors would be needed for some applications. Our current work shows that pursuit of these extensions should be fruitful.

Discrete relaxation was chosen to control tracking so that we could easily experiment by switching on or off sources of information and develop our software in a modular way. Moreover, the label elimination approach easily represents the ambiguity occurring in real-life applications. The key to reducing the computational requirements is to eliminate many labels at each filtering step while keeping those labels compatible with observation. If there are N objects and N labels, the computational complexity of tracking is potentially of the order N^2 across just two time steps. We need to continue to develop our system to perform the lower level token combination and to test it fully using a set of objects with some typical behavior. We will also make the revisions that allow objects to appear and disappear from the surveyed workspace. Also much of what has been discussed assumed objects were single independently tracked points. Clearly, some objects would be a rigid aggregate of points. For example, a truck might have a single RFID tag and perhaps four or eight visual markers that would reduce combinatorics and enable rigid motion analysis. Such planar rigid structures and symmetries are also helpful to track moving objects with wide variations in position and orientation.

References

[1] *Active RFID and Sensor Networks 2011-2021*, http://www.idtechex.com/research/ reports/active-rfid-and-sensor-networks-2011-2021-000255.asp, Accessed August 22, 2013.

[2] H. Chae and K. Han, "Combination of RFID and Vision for Mobile Robot Localization," *Proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information Processing Conference*, pp. 75-80, Dec. 2005.

[3] *Convergence System Ltd. RTLS Development Kit*, http://www.convergence.com.hk/rtls-development-kits/, Accessed August 22, 2013.

[4] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting Moving Objects, Ghosts, and Shadows in Video Streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1337-1342, Oct. 2003.

[5] *Dallas Zoo Tracks Elephants using CSL Real Time Location System*, http://rfid.net/news/399-dallas-zoo-

track-elephants-real-time-location-system, Accessed August 22, 2013.
[6] D. Hanny, M. Pachano, and L. Thompson, *RFID Applied*, John Wiley & Sons, Hoboken, N. J., 2007.
[7] H. Hontani, M. Nakagawa, T. Kugimiya, K. Baba, and M. Sato, "A Visual Tracking System using an RFID-Tag," *Proceedings of SICE Annual Conference*, pp. 2720-2723, Aug. 2004.
[8] *How Passports Work*, http://www.howstuff works.com/passport.htm, Accessed August 22, 2013.
[9] *How to Install a RTLS*, http://rfid.net/basics/rtls/241-how-to-install-a-real-time-location-system-rtls, Accessed August 22, 2013.
[10] S. Jia, J. Sheng, and K. Takase, "Obstacle Recognition for a Service Mobile Robot Based on RFID with Multi Antenna and Stereo Vision," *Proceedings of the IEEE International Conference on Information and Automation*, pp. 125-130, June 2008.
[11] C. Lin, W. Peng, and Y. Tseng, "Efficient In-Network Moving Object Tracking in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, 5(8):1044-1056, Aug. 2006.
[12] T. Meier and K. Ngan, "Automatic Segmentation of Moving Objects for Video Object Plane Generation," *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):525-538, Sept. 1998.
[13] S. Nakagawa, K. Soh, S. Mine, and H. Saito, "Image Systems Using RFID Tag Positioning Information," *NTT Technical Review Journal*, 1(7):79-83, 2003.
[14] A. Otoom, H. Gunes, and M. Piccardi, "Feature Extraction Techniques for Abandoned Object Classification in Video Surveillance," *15th IEEE International Conference on Image Processing*, pp. 1368-1371, Oct. 2008.
[15] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan, "Beepbeep: A High Accuracy Acoustic Ranging System using Cots Mobile Devices," *Proc. ACM Conf. Embedded Networked Sensor Systems -SenSys*, pp. 1-14, 2007.
[16] R. Raza and G. Stockman, "Target Tracking and Surveillance by Fusing Stereo and RFID Information," *Proc. of SPIE 8392, Signal Processing, Sensor Fusion, and Target Recognition XXI*, 83921J, April 2012.
[17] R. Raza and G. Stockman, "Fusion of Stereo Vision and RFID for Site Safety," *Proc. of ISCA 25th International Conference on Computer Applications in Industry and Engineering*, Nov. 2012.
[18] I. Sethi and R. Jain, "Finding Trajectories of Feature Points in a Monocular Image Sequence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56-73, Jan. 1987.
[19] L. Shapiro and G. Stockman, *Computer Vision*, Prentice-Hall, Upper Saddle River, NJ, 2001.
[20] *Shell Game*, http://en.wikipedia.org/wiki/Shell_game, Accessed August 22, 2013.
[21] T. Teixeira, G. Dublon, and A. Savvides, "*A Survey of Human Sensing: Methods for Detecting Presence, Count, Location, Track and Identity*," ACM Computing Surveys, V, 1-35, 2010.
[22] *Triangulation and Trilateration*, http://en.wikipedia.org/wiki/Triangulation, Accessed August 22, 2013.
[23] C. Veenman, M. Reinders, and E. Backer, "Motion Tracking as a Constrained Optimization Problem," *Pattern Recognition*, 36(9):2049-2067, Sept. 2003.
[24] S. Zekavat, H. Tong, and J. Tan, "A Novel Wireless Local Positioning System for Airport (Indoor) Security," *Proc. SPIE 5403, Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense III*, pp. 522-533, Sept. 2004.
[25] Q. Zheng and R. Chellappa, "Automatic Feature Point Extraction and Tracking in Image Sequences for Arbitrary Camera Motion," *International Journal of Computer Vision*, 15(1-2):31-76, June 1995.
[26] Q. Zhou and J. Aggarwal, "Tracking and Classifying Moving Objects from Videos," *Proc. IEEE International Workshop Performance Evaluation of Tracking and Surveillance*, pp. 52-59, Dec. 2001.

Rana Hammad Raza received his BE Electrical from NED University and his MS degree in Computer Engineering from The Center for Advanced Studies in Engineering (CASE). He is a US Fulbright scholar and is currently working towards his PhD in the Department of Electrical and Computer Engineering at Michigan State University. He has also been working at IBM Thomas J. Watson Research Center for his graduate research. His areas of interest are object localization and tracking, site monitoring and surveillance systems.

George Stockman is Professor Emeritus in the Department of Computer Science and Engineering at Michigan State University, where he has been since 1982. He teaches programming and computer vision and does research in computer vision in the Lab for Pattern Recognition and Image Processing. He has been both Associate Chair and Acting Chair of the Department. He has a BS from East Stroudsburg University, an MAT from Harvard, an MS from Penn State, and a PhD in Computer Science from the University of Maryland. He is coauthor of the 2001 textbook Computer Vision with Linda Shapiro. Recent research projects include fusion of range and video for obstacle avoidance, person verification using 3D sensing, and fusion of RFID and video for work zone safety. In addition to his academic positions, he has worked four years in industry developing software for image analysis and design of automobile seating using human body and automobile models.

An Algorithm for Interpreting Closure Property of JavaScript through Runtime Stack (A Lightweight Interpreter for Embedded Device)

Binod Kumar Pattanayak*
Siksha 'O' Anusandhan University, Bhubaneswar, INDIA

Sambit Kumar Patra†
Comviva Technology, Bangalore, INDIA,

Bhagabat Puthal‡
Siksha 'O' Anusandhan University, Bhubaneswar, INDIA

Abstract

In JavaScript, a closure is formed by returning an embedded function object that was created within an execution context of a function call or by assigning a reference of such an object to outside variable, i.e., to a global variable, a property of a globally accessible object or an object passed by reference as an argument to the outer function call. In this paper, we deal with a runtime stack with execution stack to handle embedded function object, which is platform independent and capable of being run on various feature phone mobile devices.

Key Words: Closure property, runtime-sack for mobile device, script interpreter, JavaScript interpreter, JavaScript compiler.

1 Introduction

Closure is one of the most powerful features of JavaScript that allows inner functions, i.e., function definition inside the function bodies of another function can be accessed outside its define scope. A closure is made when one such inner function is made accessible outside the function in which it was contained, so that it can be executed even after the function has returned. At that point, it still has access to the local variables, parameters and function declaration of its outer function. Those local variables, parameters and function declarations, have the same value that they had when the outer function returned and may be interacted with by the inner function which leads to a memory leak. In this paper, we address the challenges of closure property in the JavaScript and design an algorithm to resolve this using runtime stack and execution stack without any memory garbage. As we verified this algorithm in feature phones, we have not compared the execution time and memory with other algorithms run on the smart phone.

We designed an AST based script engine that consists of script objects, compiler and interpreter. Script objects will load the JavaScript built-in objects [ECMA] into the environment and also create objects at the time of interpretation. The script compiler compiles the source script. The compiled source script is represented as an AST and Symbol Table. In JavaScript the, keyword "var" is used to define a variable either in a global scope or function scope. The compiler records the variable name, used inside the function, into the symbol table with an offset number which cannot be updated during the execution. The script interpreter interprets the AST node by using the Symbol table and with the Instruction stack (Non- recursive Stack), Execution Stack and Runtime stack. Traversing the AST will be a typical post-order traversal, which also must be implemented without recursion, for which we may use Instruction Stack. As we move from one execution context to the other, we may require to push them one after the other into a stack called Execution Stack, so that we can come back to the previous execution context with a mere pop. The idea is to emulate the way recursion really works in the existing machine architectures. It involves usage of a Runtime Stack in the Data Segment. The Runtime Stack consists of Stack Frames where each stack frame refers to a function call. Similar behavior has to be emulated in the form of a stack using linked-list; we can use the same name Runtime Stack for this.

On earlier use of the executor's runtime stack, it was limited to keeping track of intermediate values while evaluating the expression. In a complete interpreter, the runtime stack plays a much greater role. At any point during the execution of a program, the stack represents a "snapshot" of the execution state. It contains the information about the procedures and functions that have been called, their return values and addresses, the current values of all their formal parameters and local variables, and which of these values are accessible according to the scope rules [20].

In JavaScript, for each function call, the execution context stack top is incremented and runtime stack is created. Runtime stack is created based on a number of arguments and local

* ISCA Member, Department of Computer Science and Engineering, ITER. Email: bkp_iter@yahoo.co.in.
† Core Engineering Group. Email: sambit.patra@comviva.com.
‡ Department of Computer Science and Engineering, ITER. Email: bputhal@gmail.com.

variables (e.g., var a; inside function). All these local variables declared and defined inside the function have the function scope. The runtime stack is created with its entries being type with the values of arguments and local variables updated. Whenever any symbol local to function is looked up, the offset value of the symbol is looked up in the runtime stack. If the symbol is found, then the value would be returned or updated into the runtime stack. The execution stack is destroyed at the end of function call when the function returns the function value. However as per the definition of closure when inner function is assigned or returned outside of the function scope, it contains its local variable as well as all the local variable of parent function. In closure the inner function becomes an embedded function object and contains the property of parent function object. It means the execution stack should not be destroyed at the end of the function call once it encounters a closure property, which may lead to a memory leak. In JavaScript, besides return or assign, inner function can be invoked outside by using "eval", "with" and "for-in". Considering all the scenarios of closure, in this paper, we propose a generic solution to solve the closure property without any memory leaks. We discuss all the related work on closure in Section 2, the problem definition in Section 3, the structure of function object, embedded function object that is used for closure, the execution stack and runtime stack has been defined in Section 4. The closure algorithm publishing the result of experimental test cases has been defined in Section 5 and 6 and we conclude the closure in Section 7.

2 Related Work

Numerous works have been proposed on Run-time systems in the literature by various authors. Authors in [13] use a Distributed Cactus stack in order for Runtime stack sharing in the Chime Parallel Processing system that enables real shared-memory multiprocessor semantics, which is achieved by a runtime system. In addition, the runtime system is also able to provide nested parallelism, task synchronization, load balancing as well as fault tolerance. Using closure conversion, an attempt is made by the authors in [7] to bridge the gap between the functional evaluators and abstract machines for the λ-calculus. Reduction of run time of indirect branch statements is addressed in [3]. A stack based platform independent Smart Virtual Machine (SVM) is detailed in [16] that is capable of running on different smart devices. An analysis of runtime stacks manipulation by various programs on Intel x86 machines is discussed in [4], which optimally improves reasoning and manipulation of values stored on the run time stack. In order to bridge the gap of speed between the processor and memory, a stack cache memory is proposed in [8] which demonstrates a hit ratio of 99 percent. A flexible Java execution environment for mobile devices is proposed in [15], which supports dynamic adaptation to needs of the applications and available resources. A JavaScript Abstract Machine (JAM), detailed in [6] represents a precise model of stack structure and it confers precise control-flow analysis in the presence of control effects such as exceptions. Two scripting languages, namely, Yolan and LightScript proposed in [18] enable scripting on low-end mobile devices and are capable of loading and executing scripts in the source form at run time with a run time stack. Authors in [6] present a modular design of JavaScript, where the modules can be loaded from external sources and it supports a flexible mechanism for dynamically loading the code isolated from untrusted modules. An integrated compile-time and run-time system is described in [21] which contributes to enhanced efficiency of shared memory parallel programming on distributed memory systems. Run time fault location and its isolation in a multi-language program can be successfully addressed by the debugging tool proposed in [23] with a CallStack. A run time system discussed in [2] is capable of providing a variety of run time facilities such as easy implementation of code generators, quick program execution and efficient garbage collection. The cactus-stack problem is addressed in [24] using Thread-local Memory Mapping (TLLM) for re-implementation of cactus-stack in the open-source Click-5 runtime system. A symbolic stack addressing can help for fetching and storing without significantly increasing execution time, as detailed in [1]. An exclusive study on verification and analysis of symbol table for C++ compiler is described in [27].

The modern literature incorporates significant contributions on closure properties from various authors. Y. Minamide, et al. [13] investigate the typing properties of closure conversion for simply-typed and polymorphic calculi, where authors translate well-typed source codes to well-typed target codes that makes it convenient in the later phases of compilation to take advantages of types in order for representation analysis and tag-free garbage collection, and facilitates correctness proofs too. The closure properties of languages generated by linear conjunctive grammars are addressed in [14] and in specific, closure under complement and show that it is closed under all set-theoretic operations where the authors consider that the concatenation of two linear conjunctive languages is concluded to be linear conjunctive too. Closure properties of slender languages with respect to several operations are discussed in [16]. Closure properties of language families under the bio-operations are investigated by authors in [5]. D. Peled, et al. [17] propose an algorithmic approach in order for checking closure properties of temporal logic specifications and co-regular languages thereby demonstrating the existence of a wide range of equivalence relations that make determining closure decidable. Authors in [9] develop a denotational semantics for constraint logic programming languages with dynamic scheduling that incorporates a semantic based on closure operators. P. Jeavons, et al. detail the closure properties in their work [11] and justify that any set of constraints not leading to an NP-complete class, must satisfy to certain type of algebraic closure condition. Decidability of emptiness and closure properties for the class of automata accepting set constraints is proved in [10]. A compositional and inductive semantic definition in fix-point, equational, constraints, closure condition, rule based and game theoretic form is discussed in [4]. In [15] we present a non recursive

algorithm for JavaScript that we have tested and verified with top 10 Alexa web sites in different mobile devices such as Moto RAZR v3 (brew 3.15), Qtopia (Linux OS), Samsung (Windows) and Nokia Series (Symbian OS).

3 Problem Definition

In JavaScript, a function can be defined inside a function as inner function property. A function object can be assigned to a variable and it holds the function body and its local variables. If a non local variable of the function scope is assigned to the inner function, as per the closure behavior discussed earlier, the variable can be accessed out of the scope of the function as follows:

```
1.  <script>
2.  var CLOSURE ;
3.  function foo(. . .)
4.  {
5.      var A = 5;
6.      function subfoo(args)
7.      {
8.      document.write(args+A); /*Print 5*/
9.      }
10.     subfoo("inside the foo  scope");
                /*subfoo(): executed inside the function scope*/
11.     CLOSURE = subfoo;
12. }
13. foo();
14. document.write(A); /*Print undefined*/
15. CLOSURE ("out side the foo scope");
            /* subfoo(): executed outside the function scope*/
16. <script>
```

The above script interpreter executes the script in the sequence 13, 5, 10, 8, 11, 14, 15, 16. In the statement 13, the function foo() executes first. In the statement 5, "A" is a local variable of function foo() and assigns with value 5, which can be accessed through foo() scope only. In statement 10 subfoo() is called. In statement 8 executes the next that is "document.write" which prints the value of "A". It searches the value of "A" in the local scope of subfoo(), in failure it searches into its parent scope foo() and print the value "5". In statement 11 "subfoo" is assigned to 'CLOSURE'. Then next executable statement is 14. Before execution of statement 14, the interpreter closes the function foo() and clear all its scope. In statement 14, it prints "undefined" as "A" is not defined in the global scope. In statement 15, CLOSURE is called as similar as subfoo(). It fails as "subfoo" is defined in function foo() and it's scope is already closed after execution of statement 13.

However, as per the closure property the statement 15 should execute as subfoo() earlier and should print the value of "A" as 5.

Similarly, the in the following script

```
1.  <script>
2.  var CLOSURE1, CLOSURE2;
3.  function foo(. . .)
4.  {
5.      var A = 5;
6.      function subfoo(args)
7.      {
8.          A++;          /*"A" is increased to 6*/
9.          document.write(args+A);
10.     }
11.     subfoo("inside the foo  scope") ;  /*It prints "A" as 6*/
12.     CLOSURE1 = subfoo;
13. CLOSURE2= subfoo;
14. }
15. foo();
16. CLOSURE1("out side the foo scope");
                                        /*It prints "A" as 7*/
17. CLOSURE2("out side the foo scope");
                                        /*It prints "A" as 8*/
18. <script>
```

In the above code fragment, the output of value "A" will be 6, 7 and 8, i.e., after execution of statement 11, 16 and 17. Here, the value of "A" will be increased for every execution of subfoo(). In JavaScript, besides functions, this closure property appears under different features like "with object" and "eval", which makes closure more complex. Considering all the complexity of closure, in this paper, we propose a generic solution to solve the closure property without any memory leaks.

4 Structures

Function Object: A function object upon deciding to retain the runtime stacks, which contains its local variables, parameters and other function declarations, of its outer functions being under execution, it should keep the runtime stacks in its own object even after the outer functions terminate. Later, while invoking this function as a closure function, the lookup for variables should take place along this list of runtime stacks.

Function Structure: A typical structure of the function object contains: a) Function Body, the AST nodes, b) Embedded function Object List (EFO List). Figure 1 demonstrates Function Object structure.

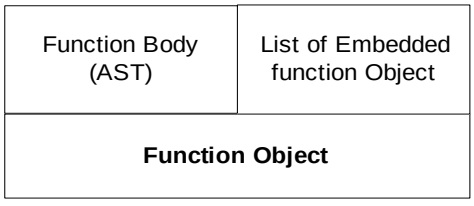


Figure 1: Function object

4.1 Embedded Function Object List (EFO)

When a function is going to terminate, it contains the list of all embedded objects inside the function. It contains: a) Runtime Data, b) Number of references to this EFO list, c) Function Block (Scope of the function of whose EFO list is made), d) Point to the next EFO list. Figure 2 depicts the EFO list structure.

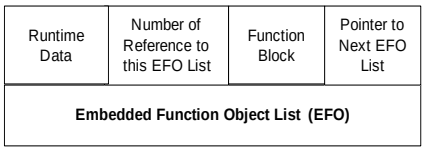


Figure 2: Embedded function object list

Runtime Data: As shown in Table 1, at the time of execution, the function object can be assigned to a variable inside a "function", "with object" or in "eval property". Thus, the runtime data contains: (a) With object/Function, (b) Eval Stack List. Figure 3 represents the Runtime Data structure.

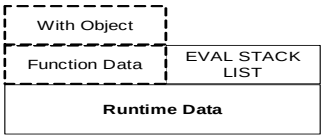


Figure 3: Runtime data

4.2 Eval Stack List

In JavaScript, the statements in the "eval" are evaluated at the runtime. When the interpreter encounters the eval statement, it sends that string to the compiler, gets its symbol tables and the AST body of the "eval" statements.

It contains: a) eval stack (offsets of variables), where the size of eval stack is set by the interpreter depending on the size of the operation stack used by an average function and the acceptable call depth; b) Eval symbol table; and c) Pointer to the next eval list (If the function has more eval statements). Figure 4 demonstrates EVAL list structure.

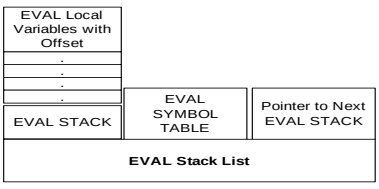


Figure 4: EVAL stack list

Function Data or With Object: Depending on the closure behavior, it may be inside a function or inside the "with" object. For "with", it is the object for function that contains a) Runtime Stack (function local variable with offset), where size of runtime stack is set by the interpreter depending on the size of the operation stack used by an average function and the acceptable call depth, b) Number of formal arguments, c) Number of actual arguments. Figure 5 represents function data.

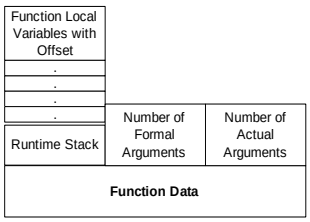


Figure 5: Function data

Execution Stack: For each function call, the execution context stack top (Figure 6) will be incremented and runtime data (Figure 3) will be created. Runtime data create function data (Figure 5) and it creates runtime stack based on number of arguments and local variables (e.g., var a; inside function).

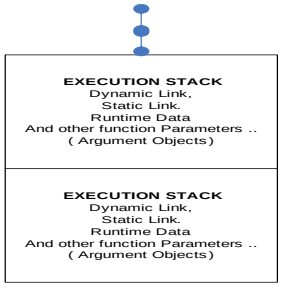


Figure 6: Execution stack

All these local variables declared and defined inside function will have the function scope. The runtime stack is a 4-byte signed integer type sequence. Whenever any symbol local to function is looked up, the value of the symbol will be looked up in the runtime stack. If the symbol is found, then the value will be returned or updated, else a new entry will be created into the global scope. The execution stack (Figure 6) will be destroyed at the end of function call when the function returns the function value.

5 Algorithm

Seven algorithms are distinguished in 6 as A, B, C, D, E, F. The functional behaviors of the algorithms are as follows.

While updating the variable from right hand (RHS) side to left hand side (LHS) in algorithm (1), it frees the occupied memory of LHS. The memory has been free if the LHS contains string, number or function objects without any embedded function object. If the LHS is a function object and having embedded function object it reduces the reference counter and free the memory if the reference counter is 0. The functional aim of the algorithm (2) is to find out whether a function should behave as a closure function or not. The condition is to choose a function to behave as a closure must be implementation dependent. Upon deciding a function to behave as a closure, the algorithm will create Embedded

A: Check for Embedded Function Object

Algorithm (1): UPDATE VALUE FROM RIGHT TO LEFT HAND SIDE VARIABLE

```
   PROCEDURE UPDATE_VALUE_FROM_RIGHT_TO_LEFT
ARG1: LHS ,
ARG2: RHS ,
   ARG3: EXEC_STACK
{
      if (NOTE_EQUAL ( LHS, RHS ) ){
   {

         if (IS_A_FUNCTION_OBJECT (LHS)  &&  EFO_LIST exist in LSH )
      {
      EFO_LIST->RefEFOCount -- ;
      If (EFO_LIST->RefEFOCount==0)
      {
        FREE (EFO_LIST) ; /*Free the EFO List*/
        FREE_VALUE ( LHS ) ; /*Free the Function Object*/

   }
   }else
   {
      FREE_VALUE ( LHS ) ; /*Free function object, string and Number*/
   }
   COPY_VALUE  ( RHS,  LHS ) ;
   if ( IS_NOT_A_LOCAL_FUNCTION_VARIABLE ( LHS ) AND IS_A_FUNCTION_OBJECT (LHS) )
   {
   CHECK_FOR_EFO (LHS, EXEC_STACK ) ;
   }
   }
}
```

Algorithm (2): CHECK FOR EMBEDDED FUNCTION OBJECT

```
PROCEDURE CHECK_FOR_EFO
ARG1: FN_OBJECT,
ARG2: EXEC_STACK
{
   if ( EFO_LIST exist in FN_OBJECT )
     {
    EFO_LIST->RefEFOCount ++ ;
   }
   else
   {
   CREATE_EFO_LIST ( FN_OBJECT , EXEC_STACK )  ;
   }
}
```

Function Object list (Discussed in B) to append the necessary runtime stack information of its outer parent functions on to it. Upon successful addition of information, the function object will contain a valid member in its structure called "Embedded function object list". This is the structure that contains all the required information pertaining to one execution level, its runtime stack, eval stack, etc. So there could be a list of more than one such structure listed depending upon the number of functions (of parent child relation) being executed at that point of time. In case the function might be already carrying such a list on it, a counter keeps the count of such reference, 'Number of Reference ', present in the Embedded function object list (Figure 2) structure, is incremented, without creating a new list. The reference counter is increased in algorithm (2). The reference value shows the number of embedded function objects assigned during the function execution. The reference counter will be reduced, whenever an object is deleted. The function object will be destroyed when the reference counter becomes 0. It will help to handle the memory garbage.

The algorithm (3) creates an entry for every execution layer

B: Create Embedded Function Object List

Algorithm (3): CREATE EMBEDDED FUNCTION OBJECT

```
PROCEDURE CREATE_EFO_LIST
ARG1: FN_OBJECT,
ARG2: EXEC_STACK
{
EMBED_FN_SCOPE: = SYM_GET_SYMBLOCK_PARENT( FN_OBJECT ) ; /*Embedding function's scope*/
   STACK_NO  :=     GET_EXECSTACK_TOP( EXEC_STACK ) ;
   while(  -- STACK_NO >= 0 && EMBED_FN_SCOPE )
   {
      /*Step #1: Find and index the private function's parent's scope in the function scope*/
      RUNTIME_DATA    :=  GET_EXECSTACK_FROM_TOP( EXEC_STACK ) ;
      if ( IS_EXECSTACK_TOP_HAS_FUN_SCOPE( EXEC_STACK, STACK_TOP ) )
      {
         EVAL_LIST  :=  GET_EVAL_LIST_FROM_RUNTIME_DATA ( RUNTIME_DATA ) ;
         /*Step #1.1 :If Eval scope present inside the function scope,Search across eval chain*/
         if (EVAL_LIST != NULL ) {
         EMBED_EVAL_SCOPE  :=  GET_MATCH_EVAL_SCOPE ( EVAL_LIST, EMBED_FN_SCOPE ) ;
      }
      /* Step #1.2 :If scope is not indexed yet, get the StackNo-level function object for match*/
      if ( EMBED_FN_SCOPE != EMBED_EVAL_SCOPE )
      {
         EMBED_EVAL_SCOPE  :=   SIP_GET_FN_SYMBLOCK ( EXEC_STACK, STACK_TOP );
      }
/* Step #1.3 :If indexed scope matches with the embedding scope, create EFO entry */
      if ( EMBED_FN_SCOPE == EMBED_EVAL_SCOPE )
      {
         /* Step #1.3.a : If matching function object already has EFO entry, inherit, else create new */
         FN_EFO_LIST   :=   ALLOCATE_MEMORY_FOR_EFO_LIST ;
         if ( FN_EFO_LIST ) {
          if ( EFO_LIST )   {
             EFO_LIST->NEXT := FN_EFO_LIST ;
             EFO_LIST    := EFO_LIST->NEXT ;
          } else {
             EFO_LIST := FN_EFO_LIST ;
          }
             REFERENCE_NUMBER   := GET_REFERENCE_NUMBER ( EFO_LIST ) ;
             REFERENCE_NUMBER   := 0 ;
             EFO_LIST->RumtimeData    := RUNTIME_DATA ;
             EFO_LIST-> FnBlock   := EMBED_FN_SCOPE ;
             EMBED_FN_SCOPE      := SYM_GET_SYMBLOCK_PARENT( EMBED_FN_SCOPE ) ;
             }
          }
      }
    /*Step #2: Find and index the private function referred(assigned)inside with object*/
      else
      {
         WITH_EFO_LIST = GET_NEW_EFO_LIST_FOR_WITH ( EXEC_STACK, RUNTIME_DATA) ;
         if ( WITH_EFO_LIST )
        {
         if ( EFO_LIST )
         {
           EFO_LIST->NEXT  := WITH_EFO_LIST ;
             EFO_LIST    := EFO_LIST->NEXT ;
       }
        else
```

```
        {
          EFO_LIST := WITH_EFO_LIST ;
        }
            /*If WITH contains one or more eval, check if 'vEmbedFnScope', i.e parent function scope*/
            EVAL_LIST := GET_EVAL_LIST_FROM_RUNTIME_DATA ( RUNTIME_DATA ) ;
            if (EVAL_LIST != NULL )
            {
            if ( GET_MATCH_EVAL_SCOPE ( EVAL_LIST, EMBED_FN_SCOPE ) )
        {
            EMBED_FN_SCOPE := SYM_GET_SYMBLOCK_PARENT( EMBED_FN_SCOPE ) ;
          }
        }
       }
    }
   }
   SIP_SET_FN_EFOLIST(  FN_OBJECT , EFO_LIST ) ;
}
```

(if any) in the execution stack, be it for a function call or a 'with' object, if the outer function is parent of the inner function. Hence, for every execution layer from top to empty, the child and parent relationship is maintained by creating link list. If the outer execution layer refers to a 'with' object, then node entry is created with a slightly different content with the help of algorithm 'create with object' (Discussed in C). Since an execution layer may refer to both a function call and a 'with' object, the variable 'Runtime Data' to this structure, which in turn may accommodate both function's runtime stack in case of function layer, or the 'with' object if it belongs to 'with' object layer. Structure "Runtime Data" also contains any eval stack present inside that function or 'with' layer. Every time a function becomes closure, it keeps its surrounding outer function runtime stacks. Hence, the same runtime stack information of outer functions are shared if more than one inner function becomes closure.

C: Match Eval Scope

Algorithm (4): MATCH EVAL SCOPE

```
PROCEDURE  GET_MATCH_EVAL_SCOPE
ARG1: EVAL_LIST,
 ARG2: SCOPE_TO_MATCH

{
 while( EVAL_LIST )
{
        EVAL_SCOPE := GET_EVAL_SCOPE ( EVAL_LIST ) ;
      if ( SCOPE_TO_MATCH  == EVAL_SCOPE )
{
          return EVAL_SCOPE ;
        }
        EVAL_LIST :=  EVAL_LIST-> NEXT ;
 }
 return NULL ;
}
```

The algorithm (4) is used to find out whether a scope is the same as one of the eval scopes in an eval list. It will compare the input scope with the eval scopes present in the input eval handle.

D: Embedded Function Object List Creates With Object

Algorithm (5): EFO CREATES IN WITH OBJECT

```
PROCEDURE GET_NEW_EFO_LIST_FOR_WITH
ARG1: RUNTIME_DATA
{
 EFO_LIST = ALLOCATE_MEMORY_FOR_EFO_LIST ;
 if ( EFO_LIST )
 {
   EFO_LIST->RumtimeData  :=  RUNTIME_DATA ;
   EFO_LIST->bIsObject        := E_TRUE ;
   EFO_LIST->RefEFOCount : = 0 ;
   EFO_LIST->FnBlock          := NULL ;
   return EFO_LIST ;
 }

 return NULL ;
}
```

This algorithm (5) creates closure (EFO) entry when the execution layer refers to a 'with' object.

The algorithm (6) is used to look up a variable in the Embedded Function Object list of a function object. It searches in all the entries in the list and returns the value if found, unsuccessful report otherwise. This algorithm will search every Embedded Function Object entry depending upon whether it is a "function call" entry or 'with' entry, by handling it in separate ways, and goes to the next entry if not found. For a function call Embedded Function Object entry, the algorithm is discussed in E to retrieve the value.

E: Get Value from the Embedded Function Object List

Algorithm (6): GET VALUE FROM THE EMBEDDED FN OBJECT

```
PROCEDURE GET_VALUE_FROM_EFO_LIST
ARG1: VARIABLE_NAME ,
 ARG2: EFO_LIST
{
while ( EFO_LIST )
 {
   RUNTIME_DATA     :=     EFO_LIST ->RumtimeData ;
   /**EFO INSIDE WITH OBJECT**/
   if (  E_TRUE == EFO_LIST ->bIsObject )
   {
     /* Step #1.1 Search in the Object*/
      if ( ! IS_GET_VALUE_FROM_WITH_OBJECT (RUNTIME_DATA , VARIABLE_NAME , &VARIABLE_VALUE ) )
      {
       /* Step #1.2 If not found above, check in the eval scope if present*/
       if ( !  IS_GET_VALUE_FROM_WITH_EVAL_LIST ( RUNTIME_DATA ,  VARIABLE_NAME ,  &VARIABLE_VALUE  ) )
       {
         EFO_LIST := EFO_LIST -> NEXT;
       }
       else  {
            return VARIABLE_VALUE  ;
       }
     else  {
       return VARIABLE_VALUE  ;
     }
   }
   else  /**EFO INSIDE WITH FUNCTION SCOPE**/
   {
     /** Step #1.1 First Check In Eval Scope/Scopes */
     if ( ! IS_GET_VALUE_FROM_FUNCTION_EVAL_LIST ( RUNTIME_DATA , VARIABLE_NAME , &VARIABLE_VALUE  ) )
     {
       /* Step #1.2 Search in the Function scope present in the EFO entry */
       EMBED_FN_SCOPE  :=     EFO_LIST->FnBlock ;
       if( ! IS_GET_OFFSET_VALUE_FROM_EMEBED_SCOPE
                                    ( EMBED_FN_SCOPE,VARIABLE_NAME ,  &OFFSET_VALUE  ) )
       {
         EFO_LIST  :=      EFO_LIST -> NEXT;
       }
       else  {
         VARIABLE_VALUE  :=  GET_VALUE_FROM_EFO ( RUNTIME_DATA , OFFSET_VALUE  ) ;
         return VARIABLE_VALUE  ;
       }
     }
     else
     {
       return VARIABLE_VALUE  ;
     }
   }
 }
 return NULL ;
}
```

F: Get Value from the Embedded Function Object Entry

Algorithm (7): GET VALUE FROM THE EMBEDED FN OBJECT

```
PROCEDURE GET_VALUE_FROM_EFO
ARG1: RUNTIME_DATA ,
ARG2: OFFSET_VALUE
{
  /*Get Variable value in the function runtime stack*/
 RUNTIME_STACK   :=  GET_CURRENT_RUNTIMESTACK
                     ( RUNTIME_DATA ) ;
 VARIABLE_VALUE  :=
                  RUNTIME_STACK + OFFSET_VALUE  ;
 return VARIABLE_VALUE ;
}
```

The algorithm (7) is used to get a variable in a function from its runtime stack present in the Embedded Function Object entry structure. It retrieves a value from the runtime stack with the help of offset. We are accessing the algorithms using the following closure example.

For example:

Similarly, the in the following script

```
1.  <script>
2.  var CLOSURE1 = "ABC", CLOSURE2="XYZ";
3.  function foo(. . .)
4.  {
5.      var A = 100;
6.      function subfoo(args)
7.      {
8.       A++;          /*"A" is increased to 101*/
9.       document.write(args+A);
10.     }
11.     subfoo("inside the foo  scope") ;
                                    /*It prints "A" as 101*/
12.    CLOSURE1 = subfoo; /*Algorithm 1, 2, 3*/
13.   CLOSURE2= subfoo; /*Algorithm 1, 2, 3*/
14. }
15. foo();
16. CLOSURE1("out side the foo scope")
                                      /* Algorithm 6, 7 */
17. CLOSURE2("out side the foo scope");
                                      /* Algorithm 6, 7 *//
18. CLOSURE1 = 10 ;          /*Algorithm 1: it reduces the
            reference count of EFO and function object */
19. CLOSURE2 = 20 ;   /*Algorithm 1: it frees the memory
                            of EFO and function object */
20. <script>
```

While executing statement 12, as per the algorithm 1, 2 and 3, the variable CLOSURE1 will free the earlier string value and the function object of subfoo() has been copied to CLOSURE1. It creates a new embedded function object as it is a closure property. In statement 13, the function object of subfoo() has been copied to CLOSURE2 similarly CLOSURE1, however it increases the reference counter of embedded function object as it is already created earlier. During the execution of statement 16 and 17, the value of "A" prints as 102 and 103 as per the algorithm 6 and 7. In statement 18, the reference counter of function object will be reduced as CLOSURE1 has been assigned to numeric value 10. The function object will be freed completely during the execution of statement 19 i.e., CLOSURE2 = 20, as the reference counter of function object is zero.

6 Experimental Testing

We have downloaded the test scripts of ECMA objects from OMA-ESMP test cases (Open Mobile Alliance-ECMA Script Mobile Profile) and verified the algorithms. The results are expected as per the test cases. We have also tested and verified this algorithm with top10 Alexa web-sites in different feature phones. It executes all the scripts of the web-sites without blocking any mobile operation. We have ported, tested and verified our script engine with low end devices such are Moto RAZR v3 (brew 3.15), Qtopia (Linux OS), Samsung (Windows) and Nokia Series (Symbian OS).

We prepare some typical closure scenarios from different website. Table 1 describes the list of typical closure behavior test cases of JavaScript. We have verified these test cases by comparing with different browsers like "Chrome", "Mozilla" and "IE".

7 Conclusion

During the execution of the function, the values of the local variables are stored into the runtime stack according to their offset values from the symbol table. This algorithm will keep the runtime stack into the function object while accounting closure property during the execution. Here the reference counter plays the crucial role to handle the memory garbage. The reference counter will increase when an embedded function object is assigned to the variable and will be reduced when an object is deleted. The runtime stack will be deleted when the reference count is reduced to zero. For looking up a symbol, it is required to lookup first into the local symbol table and then eval scope or with object and global symbol table. In the future, we need to optimize the algorithm further to reduce the number of lookup, execute the algorithm in smart phones and compare the algorithm with others to optimize further.

References

[1] T. Adin, "Symbolic Stack Addressing," *The Journal of Forth Application and Research*, 5(3):365-379, 1989.

[2] A. W. Appel, *A Runtime System*, Draft, Princeton University, February 1989.

[3] L. Cullen, D. Saumya, A. Gregory and S. Benjamin, *Stack Analysis of x86 Executables, Technical Report*, The University of Arizona, Tucson, April 2004.

[4] P. Cusot and R. Cusot, "Compositional and Inductive Semantic Definitions in Fix-Point, Equational, Constraint, Closure Condition, Rule Based and Game Theoretic Form," *Lecture Notes in Computer Science*,

Table 1: Closure test cases

Sl.	Scripts	Results	Remarks

1	function say()
	{
	var num = 6;
	var sayAlert = function()
	{
	if (num == 7)
	{
	document.write("Success")
	}
	num++;
	return sayAlert;
	}say()();
		Success	The inner function is being called by its parent function outside of its scope.

2	var cl1 = undefined;
	function test1()
	{
	var a = 45 ;
	with (this)
	{
	function x() { return a };
	cl1 = x ;
	eval("var a = 5");
	}
	}
	test1();
	document.write(cl1()) ;
		5	The inner function “x” is defined inside “wih” operator and is assigned with closure variable “cl1”. The inner function is being invoked by closure variable cl1 outside its function scope.

3	var cl2 = undefined ;
	
	function test2 (){
	var a = 45 ;
	
	function x(){return a};
	with (this)
	{
	cl2 = x ;
	eval("var a = 5");
	}
	} test2();
	document.write(cl2()) ;
		5	The inner function “x” is assigned with closure variable “cl2” inside “wih” operator. The inner function is being invoked by closure variable cl2 outside its function scope.

4	var x = new Object ;
	x.a=100;
	x.c=10;
	function fn(){
	var a = 10;
	with (x){
	eval("var a = 200;
	c=function closure(){return a;};");
	}
	}
	fn();
	document.write(x.c()) ;
		200	The inner function “closure” is defined inside ”eval” and assigned to global object “x” using “with” operator. The inner function is invoked by “x.c” which is outside of its functional scope.

5	var x = new Object ;
	x.a=100;
	x.c=10;
	function fn(){
	var a = 10;
		10	The inner function “closure” is defined and assigned to global object “x” using “with” and “eval”. The inner function is invoked by “x.c” which is outside of its functional scope.

Sl.	Scripts	Results	Remarks

	function closure(){return a};
	with (x){
	eval("c=closure");
	}
	}
	fn();
	document.write(x.c()) ;		

6	var x = new Object ;
	x.a=100;
	x.c=10;
	function fn(){
	var a = 10;
	function closure (){return a};
	with (x){ c= closuer;
	}
	} fn(); document.write(x.c()) ;
		10	The inner function "closure" is defined and assigned to global object "x" using "with" operator. The inner function is invoked by "x.c" which is outside of its functional scope.

7	var a = 56 ;
	function fn (){
	var a = 19 ;
	function closure (){return a};
	this .y = closure;
	eval("var a = 5");
	}
	fn(); document.write(y());
		5	The inner function "closure" is defined and assigned to global variable "y" using "this" operator. The inner function is invoked by "y" which is outside of its functional scope.

8	var obj = {a:2, obj:{a:3}};
	with (obj){
	var f = function () {return a;};
	}
	document.write(f()) ;
		2	The function is defined inside with scope and invoked outside the scope.

9	function fn(){
	var a=20;
	var obj = {a:10, obj:{a:3}};
	with (obj){
	function closure (){
	f = function () {return a;};
	} y= closuer;
	}
	}fn(); y(); actual=f(); document.write(actual);
		20,
		10(Mozilla)
			The inner function "closure" is defined inside an object "obj" using "with" scope and invoked outside the "with" scope.
			
			However in Mozila browser it returns its object literal value "10", while other browser returns "20" the global variable.

10	var x;
	function fn(){
	function fn1()
	{
	eval("(function () {
	closure = function(){
	document.write(' fn called');
	fn2 ();};
	})();");
	}
	with(this) { fn1 (); }
	} fn(); closure ();
	fn = 5;
	function fn2 (){
	document.write("fn2 called");
	}
		fn called
		
		fn2 called
			The inner function "closure" is assigned inside eval() and is invoked outside the functional scope.

939:293-308, 1995.
[5] M. Daleya, O. H. Ibarra and L. Kari, "Closure and Decidability Properties of some Language Classes with Respect to Ciliate Bio-Operations," *Theoretical Computer Science*, Elsevier, Essex, UK, 306:19-38, 2003.
[6] H. David, *Modules for JavaScript Simple, Compatible, and Dynamic Libraries on the Web*, Draft, Mozilla Research Foundation, May 2011.
[7] V. H. David and M. Matthew, "Pushdown Abstractions of JavaScript," arXiv: 11094467[cs.PL], arXiv.org, 2011.
[8] M. Anton Ertl and D. Gregg, "The Structure and Performance of Efficient Interpreters," *Journal of Instruction-Level Parallelism*, 5(2003):1-25, 2003, (http://www.jilp.org/vol5/v5paper12.pdf).
[9] M. Falaschi, "Constraint Logic Programming with Dynamic Scheduling: A Semantics Based on Closure Operators," *Information and Computation*, Academic Press, Duluth, MN, USA, 137:41-67, 1997.
[10] R. Gilleron, S. Tison and M. Tommasi, "Set Constraints and Automata," Information and Computation, 149:1-41, 1999.
[11] P. Jeavons and D. Cohen, "Closure Properties of Constraints," *Journal of the ACM (JACM)*, ACM, New York, 44(4):527-548, USA, 1997.
[12] S. A. Mads, B. Dariusz, D. Oliver and M. Jan, "A Functional Correspondence Between Functional Evaluators and Abstract Machines," *Proceedings of the Fifth ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP'03)*, ACM Press, pp. 8-19, August 2003.
[13] Y. Minamide, I. G. Morissett, and R. Harper, "Typed Closure Conversion," POLP '96 *Proceedings of the 23rd ACM SIGPLAN SIGACT Symposium on Principles of Programming Languages*, ACM, New York, USA, pp. 271-283, 1996.
[14] A. Okhotin, "On the Closure Properties of Linear Conjunctive Languages," *Theoretical Computer Science*, Elsevier, Essex, UK, 299:263-285, 2003.
[15] S. Patra, B. K. Pattanayak, and B. Puthal, "JavaScript Interpreter Using Non Recursive Abstract Syntax Tree Based Stack," *American Journal of Applied Sciences*, 10(4):403-413, 2013.
[16] G. Paun and A. Salomaa, "Closure Properties of Slender Languages," *Theoretical Computer Science*, Elsevier, Essex, UK, 120(2):293-301, November 1993.
[17] D. Peled, T. Wilke and P. Wolper, "An Algorithmic Approach for Checking Closure Properties of Temporal Logic Specifications and Co-regular Languages," *Theoretical Computer Science*, Elsevier, Essex, UK, 195:183-203, 1998.
[18] J. Rasmus, "Design and Implementation of a Scripting Language for Mobile Devices," http://lightscript.net, 2009.
[19] R. Romansky and Y. Lazarov, "Stack Cache Memory," *Journal of Information, Control and Management Systems*, 2(1):65-74, 2004.
[20] Mak Ronald, *Writing Compilers and Interpreters*, John Wiley and Sons, Inc., 1996.
[21] D. Sandhya and R. Ramakrishnan, "Combining Compile-Time and Run-Time Support for Efficient Software Distributed Shared Memory," *Journal of Computer Science and Technology*, 16(3):231-241, 1999.
[22] S. Shantanu, M. Donald and D. Partha, "Distributed Cactus Stacks: Run Time Stack Sharing Support for Distributed Parallel Programs," *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98)*, pp. 1-5, July 1998.
[23] S. Shende, A. D. Malony, and J. C. Linford, "Isolating Runtime Faults with Callstack Debugging using TAU," *Proceedings of the IEEE High Performance Extreme Conference (HPEC'12)*, 16th Annual HPEC Conference, Waltham, USA, September 2012.
[24] I. Ting, L. Angelina, B. Silas, Z. H. Wickizer and E. L. Charles, "Using Memory Mapping to Support Cactus Stacks in Work-Stealing Runtime Systems," *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques*, ACM PACT'10, pp. 411-420, 2010.
[25] I. Ting, L. Angelina., B. Silas, Z. H. Wickizer, and E. L. Charles, "Using Memory Mapping to Support Cactus Stacks in Work-Stealing Runtime Systems," *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques*, ACM PACT'10, pp. 411-420, 2010.
[26] S. L. Yang and S. S. Yun., "A Study on the Smart Virtual Machine for Executing Virtual Machine Codes on Smart Platforms," *International Journal of Smart Homes*, 6(4):89-92, October 2012.
[27] S. L. Yang and S. S. Yun, "A Study on Verification and Analysis of Symbol Tables for Development of the C++ Compiler," *International Journal of Multimedia and Ubiquitos Engineering*, 7(4)174-186, 2012.

Binod Kumar Pattanayak completed his M.S. in Computer Engineering from Kharkov Polytechnical Institute, Kharkov, Ukraine in 1992, and Ph. D. in Computer Science and Engineering in 2011 from Siksha 'O' Anusandhan University, Bhubaneswar, India. Currently, he is placed as an Associate Professor and Head in the Department of Computer Science and Engineering, Institute of Technical Education and Research under Siksha 'O' Anusandhan University, Bhubaneswar, India. He has more than 30 research publications to his credits in reputed international journals and conferences. He has a teaching experience of more than 20 years. More than 15 students have successfully completed their M. Tech. Dissertation work under his supervision and

more than 10 scholars are pursuing their Ph.D. Dissertation work under his guidance currently. He has also acted as a visiting faculty member to different institutions within the country as well as abroad. His research areas include Ad Hoc Networks, Compilers, Software Engineering and Intelligent Systems.

Sambit Kumar Patra completed Master in Computer Application from National Institute of Electronics and Information Technology, New Delhi, India in 2004, and M.Sc. in Telecom Technology from Manipal University, India in 2009. Currently, he is Pursuing Ph.D. in Compiler and Interpreter for embedded device at SOA University, Bhubaneswar, India. Moreover, now he is working as an Associate Manager in Accenture Service Pvt Ltd., India. Prior to this, he was working as a Senior Technical Lead in Mahindra Comviva (June-2004 to Nov-2013), Bengaluru. He has to his credit as many as three research publications in reputed international journals. His research areas are Mobile and Telecom Technology, Mobile Application in Cross Platform, Compiler designing, Artificial Intelligence, Robotic and ad-hoc network.

Bhagabat Puthal (photo not available) started his career as an Asst. Professor in NIT, Rourkela from 1963 – 1981 for 18 years. Later he joined as Principal, Research Manager, Steel Authority of India Limited in 1981 and worked for 11 years. Then he worked as Professor, IGIT, Talcher from 1992 – 1997. Lastly Dr.Puthal joined ITER, SOA University, Bhubaneswar in the year 1997 and worked till 2011. Dr. Bhagabat Puthal was awarded PhD degree in, Microprocessors & Microcomputers from Sambalpur University, M.Tech from Indian Institute of Technology, Delhi (1979 – 1981) and B.Tech from Jadavpur University (1959 –1963). He has to his credit a number of research publications in reputed national and international journals and conferences. A number of research scholars have pursued their Ph. D. research dissertation works under his guidance.

Feature Selection and Decision Fusion for Distributed Classification

Hoa Dinh Nguyen*
Posts and Telecommunications Institute of Technology, Hanoi, VIETNAM, 100000

Qi Cheng†
Oklahoma State University, Stillwater, OK 74078 USA

Abstract

Classification has long been considered as an important research problem over the past decades. In this paper, we propose a new distributed classification framework for an engineering system. The proposed classification system consists of multiple local 1-rest binary classifiers and a binary decision fusion center. In the local binary classifier, a new feature selection method based on conditional mutual information is adopted to improve the classification performance as well as to reduce the computational complexity. At the fusion center, an optimal binary decision fusion rule is derived based on the minimum mean decision cost criterion. By comparing the achieved minimum cost with a threshold, we can determine 1) if the system works under an unknown condition; 2) if not, which known class the system condition belongs to. The proposed method is evaluated using both synthetic and real datasets. Experimental results show that the distributed classification system combined with the new feature selection method outperforms the existing methods. The new classification framework is a more flexible and suitable solution for both known and unknown classes.

Key Words: Feature selection, conditional mutual information, cost minimization, binary decision fusion, distributed classification.

1 Introduction

Classification problems have been considered as an important research topic for many years. There are a wide range of applications related to classification, such as character recognition, speech recognition, remote sensing, medical applications, image processing, and many others [26]. There exist various classification methods such as Bayes classifier [28], *k*-nearest neighbors (k-NN) classifier [3], neural network based classifier [27], etc. In an engineering system determining the current working state of the system can also be regarded as a classification problem.

With the rapid development of sensing technology, different kinds of sensors are now available and can provide a wide range of information for processing. Thus, we generally have to deal with a large amount of data collected from a network of distributed sensors to decide on the system condition. In statistical model based classification, the number of model parameters generally increase with the number of features the sensors measure, while available training data is limited resulting in the so called "curse of dimension" problem [4]. Furthermore, it has been shown that redundant and/or irrelevant features may severely affect the accuracy of learning methods [19]. To deal with high-dimensional data directly not only increases the computational complexity, but also brings implementation and accuracy problems. Therefore, feature selection is highly desirable since it can reduce the dimensionality and computation storage requirements, hence, facilitating data understanding and improving the classification performance.

Feature selection is a process to select a feature subset by removing both irrelevant and redundant features, which may have negative effects on classification [13]. The computational complexity of feature selection and classifier design highly depends on the number of features involved. Besides, communicating all sensing data to a processing center may be costly especially for wireless sensor networks. Therefore, a distributed feature selection and classification framework in which feature selection is conducted locally to facilitate local classification, and then decision fusion is performed to generate a global decision is a viable solution.

In *m*-ary classification problems, there are two kinds of approaches. One directly utilizes an *m*-ary classifier. The other uses multiple binary classifiers, and binary decisions are combined to generate a final classification decision. The second approach is used widely because the computational complexity for designing a binary classifier is much less than that for an *m*-ary classifier. It has been shown that for independent and identical sensors collecting a large number of short messages provides more information than a smaller number of relatively long messages [21]. Using binary classifiers is also preferable in distributed processing due to the flexibility and the ease of applying various decision fusion rules for binary decisions.

In general, there are two types of binary classifiers for *m*-ary classification problems: 1-rest classifier and 1-1 classifier. 1-rest classifier distinguishes one class from the rest. 1-1 classifier, also known as binary pairwise classifier, discriminates classes in pairs. For instance, as shown in our

* E-mail: hoadn@okstate.edu.
† School of Electrical and Computer Engineering. Phone/Fax: (405)744-9919/9198. E-mail: qi.cheng@okstate.edu.

last experiment, there are five classes: Normal, DoS, R2L, U2R and Probe. A typical 1-rest classifier can be the one to decide if the data is Normal or not, while a 1-1 classifier can be the one to decide if the data is DoS or U2R. Tax and Duin investigate the efficiency of these two types of binary classifiers and show that 1-1 classifier performs worse than 1-rest classifier [23]. This is because 1-1 classifier is trained to classify the data into one of the classes in the pair. The output is not reliable since the true class may not be either one. Hence, 1-rest classifier is preferred and widely used. In this paper, we adopt 1-rest binary classifiers in the distributed framework. Combining local binary decisions to make a final classification decision is studied previously in [16, 25, 29] where an ensemble of multilayer perceptron neural network based binary classifiers are adopted locally. It requires both positive and negative scores of all classifiers at the fusion center for decision making, which may incur high communication costs if these classifiers and the fusion center are not co-located.

The contributions of this paper are twofold. First, we develop a new feature selection method based on conditional mutual information to reduce the computational complexity as well as to enhance the accuracy of local decisions. This method can find the optimum feature subset in terms of maximizing the relevance to the class label, while minimizing the redundancy among features. The proposed feature selection approach is different from previous works [2, 12, 18] as it directly uses conditional mutual information to select features that are relevant to the class variable. Additionally, a redundancy evaluation process is integrated to eliminate any emerging redundant features in the subset. Second, we derive an optimum decision fusion rule for local 1-rest binary decisions in a Bayesian framework. All local binary decisions are combined at the fusion center in such a way to minimize the average cost of decision making. The advantage of our decision fusion method is that it can be applicable to all kinds of binary classifiers which only need to send local decisions to a fusion center. The parameters in the fusion rule are the performance measures of these binary classifiers, which can be learned from the training process. By constraining the probability of misclassifying a known class as an unknown class, a threshold is set on the achieved minimum decision cost for novel class detection. Experimental results on different types of datasets show that our new feature selection method can help improve the performance of local binary classifiers compared with existing feature selection methods, and the proposed binary decision fusion rule outperforms the use of *m*-ary classifiers as well as other previous approaches to solve for classification problems.

The remainder of the paper is organized as follows. The problem of distributed classification is formulated in Section 2. Related background information is presented in Section 3. Section 4 proposes a new feature selection method based on conditional mutual information. A minimum mean decision cost based fusion rule is provided in Section 5. Section 6 presents experimental results using both synthetic and real datasets. Conclusions are provided in Section 7.

2 Problem Formulation

The goal of classification problems may be to decide the working condition of an engineering system under monitoring. Assume that there are N types of known working conditions forming a total of N classes. N 1-rest binary classifiers are applied and their decisions are combined to determine, based on the network observations, if the system condition belongs to any of these classes. Let e_n denote the binary decision of classifier n, $n = 1, ..., N$. $e_n = 1$ if the network condition is *of class n* and $e_n = 0$ if it is *not of class n*.

As discussed in Section I, very often multiple sensors can provide information regarding the system conditions. These sensors can be distributed in the system or co-located but providing observations of different aspects of the system. Considering the potential communication cost and/or computational complexity, distributed processing is more desirable than centralized processing. Since we adopt 1-rest binary classifiers, not every sensor may be useful in distinguishing if the network condition belongs to a particular class or not. Neither do all the feature measurements in each sensor contribute the same level of information to each classifier. To improve the accuracy of designed classifiers, feature selection should be conducted to select the most informative features for each classifier. In this way, we can avoid unnecessary data processing, improve learning efficiency as well as reduce the complication of the system in the real-time decision-making process. In fact, this is one of the major motivations for the consideration of multiple 1-rest binary classifiers at each local sensor rather than a conventional *m*-ary classifier in distributed processing, since different subsets of features now can be selected for different binary classifiers.

Assume that there are K sensors in the system. Let $\boldsymbol{S_k}$ be all the feature measurements of sensor k, $k \in \{1, ..., K\}$. Let $g_n^k(\bullet)$ be the classifier of class n at sensor k. $\boldsymbol{S}_n^k \subseteq \boldsymbol{S_k}$ is the subset of features selected for $g_n^k(\bullet)$. The output $e_n^k = g_n^k(\boldsymbol{S}_n^k) \in \{0, 1\}$. If classifier n needs not to be designed at sensor k, then e_n^k does not exist. All local binary decisions are sent to a fusion center where a global decision regarding the system condition is made. Let $g(\bullet)$ be the fusion rule. Then, the global decision $G = g(\{e_n^k\}) \in \{1, ..., N+1\}$. $G = N+1$ means that the system condition does not belong to any of the known classes, i.e., it may be a defect situation or a new kind of working condition. The record that is classified as an unknown class will be stored for further investigation. Figure 1 illustrates the structure of the proposed distributed classification system with multiple sensors and multiple binary classifiers.

In this paper, the following issues are addressed. For each type of binary classifier, the most informative sensors and feature subsets need to be selected (What are sets $\Re_n$ and $\boldsymbol{S}_n^k$ for $n = 1, ..., N, k \in \Re_n$). For each sensor and selected

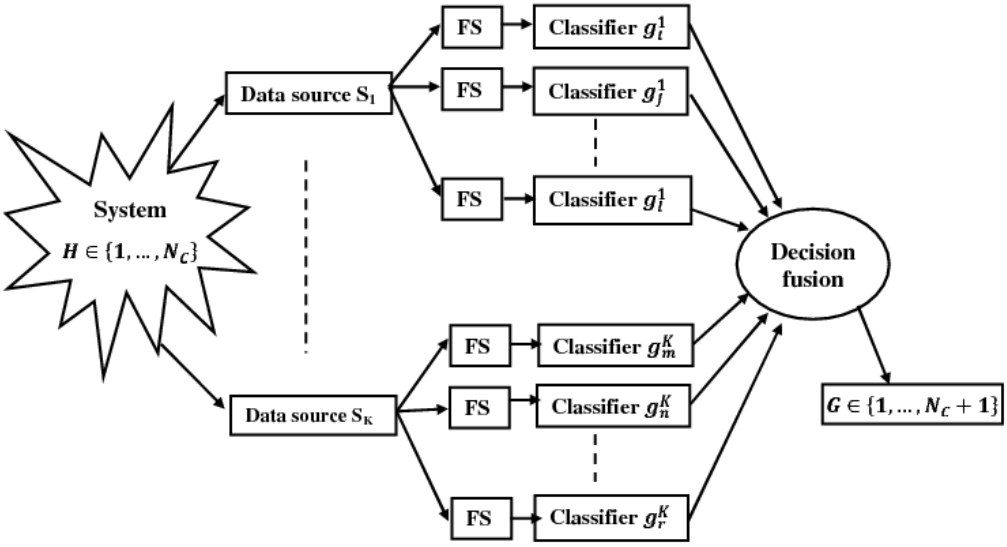


Figure 1: A multisensor - multiclassifier framework for distributed classification. An engineering system may be in one of N working states or an unknown abnormal state. Measurements related to the system are collected from K distributed sensors. At each local sensor, multiple 1-rest binary classifiers are implemented to determine if the system is in one of the working states or not. For each binary classifier design, feature selection is integrated individually. All the local binary decisions are fused at a fusion center to determine the state of the system

feature subset, local classifier n should be designed accordingly (What are $g_n^k(\bullet)$'s?). At the fusion center, an appropriate fusion rule needs to be designed to optimize the system performance (What is $g(\bullet)$?). Before these issues can be studied, we present related background information in the following section.

3 Preliminaries

3.1 Mutual Information and Classification Error

Shannon's information theory [20] formalizes the quantization of the relationship between different variables as well as the uncertainty of each variable. The uncertainty of the output values of a variable X is measured by the entropy:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \tag{1}$$

where $p(x) = P(X = x\}$, $x \in \mathcal{X}$ is the probability mass function of X. If we have already known one variable Y, the uncertainty of variable X is presented by the conditional entropy:

$$H(X|Y) = -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x|y) \log_2 p(x|y) \tag{2}$$

where $p(x|y)$ is the conditional mass probability of variable X given Y. The mutual information (MI) of two random variables is a quantity that measures the mutual dependence of those variables. This is the amount of the uncertainty of one variable being reduced by giving the knowledge of the other. MI between two discrete random variables X and Y is defined as

$$I(X;Y) = I(Y;X) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{3}$$

The MI is symmetric with respect to X and Y. The larger the MI between two variables is, the closer their relationship is. The relationship between MI and entropy is presented by the following equation.

$$I(X;Y) = H(X) = H(X \mid Y) \tag{4}$$

It has been shown in [7, 9] that the Bayes classification error is lower bounded as follows,

$$P_e \geq \frac{H(C) - I(C;\boldsymbol{S}) - 1}{\log_2(N-1)} \tag{5}$$

where N is the number of classes; $\boldsymbol{S}$ is the subset of features used for classification. $H(C)$ is the Shanon entropy of class variable, and $I(C;\boldsymbol{S})$ is the MI between class variable and the set of selected features. It is generally difficult to quantify directly the performance improvement of a classifier when selecting features. Therefore, MI is evaluated since increasing the mutual information can reduce the lower bound of the classification error in (5) for given $H(C)$.

3.2 Probability Density and Mutual Information Estimation

In fact, it is often useful to present the MI of two random variables conditioned on a third. The conditional mutual information $I(C; f_i \mid \boldsymbol{S})$ can be expressed in terms of conditional entropies as follows,

$$I(C; f_i \mid \boldsymbol{S}) = H(C \mid \boldsymbol{S}) - H(C \mid \boldsymbol{S}, f_i) \tag{6}$$

where $H(C|\boldsymbol{S})$ is the conditional entropy of class variable C given the selected features $\boldsymbol{S}$. The conditional entropy can be approximated as

$$\widehat{H}(C|\boldsymbol{S}) = -\sum_{j=1}^{M}\sum_{c=1}^{N_C} p(c)\hat{p}(\boldsymbol{s}_j|c)\log\frac{\hat{p}(\boldsymbol{s}_j|c)p(c)}{\sum_{k=1}^{N_C}\hat{p}(\boldsymbol{s}_j|k)p(k)} \tag{7}$$

where $\boldsymbol{s}_j$ is the jth training sample, M is the total number of training samples, $p(c)$ is the prior probability of class c.

The conditional mutual information estimation includes two steps: first, the probability density functions (pdfs) of related variables are estimated and then the mutual information is calculated.

The pdfs can be estimated using the Parzen window method [17]. Let vector $\boldsymbol{z}$ denote the set of selected features. Given m_c samples of $\boldsymbol{z}$ belonging to class c, the approximated conditional pdf $\hat{p}(\boldsymbol{z} \mid c)$ using the Parzen window method is

$$\hat{p}(\boldsymbol{z}|c) = \frac{1}{m_c}\sum_{i\in I_c}\phi(\boldsymbol{z} - \boldsymbol{z}_i, h) \tag{8}$$

where $c = 1, ..., N_c$; I_c is the set of indices of training samples $\boldsymbol{z}_i$ belonging to class c; $\phi(\bullet)$ is the window function; h is the window parameter. The estimation of probability density function $\hat{p}(\boldsymbol{z} \mid c)$ can converge to the true one if $\phi(\bullet)$ and h are properly chosen [17]. In the literature, a Gaussian window is usually used.

$$\phi(\bar{\boldsymbol{z}}, \boldsymbol{h}) \frac{1}{(2\pi)^{\frac{D}{2}} h^D |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{\tilde{\boldsymbol{z}}'\Sigma^{-1}\tilde{\boldsymbol{z}}}{2h^2}\right) \tag{9}$$

where $\tilde{\boldsymbol{z}} = \boldsymbol{z} - \boldsymbol{z}_i$; D is the dimensions of sample $\tilde{\boldsymbol{z}}$; Σ is the covariance matrix of $\tilde{\boldsymbol{z}}$. In order to simplify the pdf estimation for high dimensional data, "kernel independence" is assumed [11]. The general pdf estimate of a D-dimensional variable is then expressed as follows,

$$\hat{p}(\boldsymbol{z} \mid c) = \frac{1}{m_c}\sum_{i\in I}\sum_{d=1}^{D}\phi(z^{(d)} - z_i^{(d)}, \boldsymbol{h}_d) \tag{10}$$

h_d in each dimension can be chosen as $h_d = 1.06\sigma^d mc^{-\frac{1}{5}}$, where σ^d is the standard deviation of training data in dimension d [11].

4 Local 1-rest Binary Classifier and Feature Selection

The proposed framework consists of local binary classifiers and their individual feature selector, and the fusion center. They are discussed in this and the next sections.

4.1 Local Binary Classifiers

Based on the feature measurements at sensor k, a Bayesian binary classifier is adopted to distinguish class n from the rest as follows,

$$l_n^k = \frac{P(H_n|\boldsymbol{S}_n^k)}{P(H_{0n}|\boldsymbol{S}_n^k)} = \frac{P(\boldsymbol{S}_n^k|H_n)P(H_n)}{P(\boldsymbol{S}_n^k|H_{0n})P(H_{0n})}\begin{cases}\geq 1, e_n^k = 1\\ < 1, e_n^k = 0\end{cases} \tag{11}$$

where H_n denotes class n while H_{0n} denotes *not* class n. The posterior probability $P(H_n \mid \boldsymbol{S}_n^k) = P(\boldsymbol{S}_n^k \mid H_n)P(H_n)$. The prior probability of each class is assumed to be uniformly distributed, i.e., $P(H_n) = \frac{1}{N}$ and $P(H_{0n}) = 1 - \frac{1}{N}$. The prior probability can also be learned from the training dataset if needed.

4.2 Feature Selection

Given the training data containing M samples, D features and the class variable C, the feature selection problem is to select from D-dimensional measurement space a subspace of d features that best characterize C [18]. The optimality criteria can be *minimal classification error* of classifiers, or *maximal dependency* between the feature subset and class labels, etc.

Exhaustive search for the best feature subset is generally computationally expensive. There have been various sequential search based approaches including best individual features, sequential forward search, sequential forward floating search, etc. Mutual information has been explored in greedy forward feature selection methods [2, 6, 12, 18]. Battiti [2] proposes an incremental search method, i.e., mutual information based feature selection (MIFS). This method selects feature f if $I(C; f) - \beta\sum_{s\in S} I(f; s)$ is the largest. That is, feature f should be selected if it maximizes the mutual information with the class C and has minimum mutual information with existing feature s in the selected subset $\boldsymbol{S}$. The main disadvantage of this MIFS method is the difficulty in setting the parameter β. Especially, when the size of S becomes large, the selection algorithm tends to select features that are less correlated to the selected features, while may also be less correlated with the class label. Kwak and Choi [12]

modify the criterion as $I(C;f)-\beta\sum_{s\in S}\frac{I(C;s)}{H(s)}I(f;s)$ where they assume class C does not change the ratio of the entropy of s and the mutual information between s and f. In [18], β is set to be $\frac{1}{|\boldsymbol{S}|}$, where $|\boldsymbol{S}|$ is the cardinality of the selected feature subset and this can partly solve the problem mentioned above. However, the wide range of mutual information values may still cause the bias [6]. To handle this issue, the normalized mutual information is introduced in [6].

The limitations of existing mutual information based feature selection methods are twofold. First, ideally, the feature with maximum conditional mutual information $I(C;f\,|\,\boldsymbol{S})$ should be selected. Existing methods approximate it by an alternative problem that maximizes $I(C;f)$ and minimizes $\sum_{s\in\boldsymbol{S}}I(f;s)$ through a weighted sum. This approximation helps facilitate the mutual information estimation since we do not have to deal with high dimensional data. However, the conditional mutual information between f and C given the knowledge of $\boldsymbol{S}$ cannot be correctly represented by a function of the MI between f and C and the MI between f and every feature in $\boldsymbol{S}$. In other words, the feature that both maximizes $I(C;f)$ and minimizes $\sum_{s\in S}I(f;s)$ is not necessarily the one that maximizes $I(C;f\,|\,\boldsymbol{S})$. Second, current method scan select relevant features for classification but cannot eliminate redundant features if already selected. There is no step to re-evaluate the redundancy of selected features when new features are added into the subset. As a result, the selected feature subset may not be the most compact one. In [8], conditional mutual information is used as a criterion, where a new feature is added to the subset if it maximizes $\left\{\min_{s\in S}I(C;f\,|\,s)\right\}$. However, the potential redundancy with respect to a group of selected features is not considered. We intend to find the most compact feature subset while maintaining the classification performance. The proposed feature selection method is described as follows.

Algorithm:

- **Step 1**: Initialization. Let $\boldsymbol{F}$ contain "set of all D features" and $\boldsymbol{S}$ an "empty set".
- **Step 2**: Repeat until $\boldsymbol{F}=\phi$.

  1) Randomly select one feature $f_i\in\boldsymbol{F}$, compute conditional mutual information $I(C;f_i\,|\,\boldsymbol{S})$; set $\boldsymbol{F}\leftarrow\boldsymbol{F}\{f_i\}$.
  2) Feature selection: if $I(C;f_i\,|\,\boldsymbol{S})>\gamma$, set $\boldsymbol{S}\leftarrow\boldsymbol{S}\cup\{f_i\}$.
  3) Validating $\boldsymbol{S}$:
     - if the cardinality $|\boldsymbol{S}|=1$ then go to 1).
     - else a) set $k=1$.

       b) repeat until $k>|\boldsymbol{S}|$

       * Compute $I\left(C;f_k\,|\,\boldsymbol{S}\{f_k\}\right)$.
       * Se $k=k+1$.
       * If $I\left(C;f_k\,|\,\boldsymbol{S}\{f_k\}\right)\le\gamma$, then set $\boldsymbol{S}\leftarrow\boldsymbol{S}\{f_k\}$; set $k=1$.

- **Step 3**: Output selected feature subset $\boldsymbol{S}$.

In fact, this feature selection method aims to select any features that are relevant to class label, but not redundant. The threshold γ is set at a reasonably small value such that it helps remove features containing pure noise. In this feature subset selection method, we integrate a redundancy evaluation step right after a new feature is selected into the subset. This step helps remove all redundant features emerging during the selection process. The evaluation process is similar to the idea of backward feature selection. However, we only need to apply it to a small set of selected features instead of the whole original feature set, which helps reduce the computational effort significantly.

In each iteration, one candidate feature is randomly selected from the remaining set. Any feature that is informative for classification is selected, thus no feature ranking is needed. The greedy process to select new features into the subset does not consider the mutual information between each candidate feature and the class label. Instead, conditional mutual information given all selected features is used to help minimize the redundancy among features in the selected feature subset.

The estimate of the pdf (8) is also used in the Bayesian classifier given in (11). Let $P_{m;n}^{k}$ and $P_{f;n}^{k}$ denote the missed detection and false alarm rates, respectively, of the local classifier n at sensor k. Given a limit number of training samples, the 10-fold validation process [15] is used to estimate the values of these two error probabilities of all local classifiers. A local classifier is considered as *useful* if it can provide higher than 50 percent accuracy on the validation dataset (better than a random choice). That is, $P_{m;n}^{k}<0.5$ and $P_{f;n}^{k}<0.5$. To facilitate the classification process and to reduce the chance of misclassification, only useful local classifiers are selected for further processing. In other words, local classifiers that perform worse than a random choice are eliminated. The fusion center will collect local decisions only from useful classifiers.

4.3 Computational Complexity of Feature Selection

The computational complexity of conditional entropy estimation according to equation (7) is proportional to M^2d, where M is the total number of training samples and d is the number of selected features in $\boldsymbol{S}$ [12]. As a result, the

computational complexity of the loop in **b)** of the validating stage 3) in **Step 2** is proportional to M^2d^2 for all d features in $\boldsymbol{S}$. The validating stage 3) is repeated at most D times for all the features. Therefore, the computational complexity of **Step 2** is proportional to M^2d^2D. The computational complexity of the whole algorithm is the same as that of **Step 2**, and is proportional to M^2d^2D. Whereas, the computational complexity of the MIFS method and all its variances is proportional to M^3dD. Note that, in most cases, the number of training samples M is larger than the number of selected features d.

5 Minimum Decision Cost Based Fusion Rule

A local binary decision e_n^k is generated by classifier $g_n^k(\bullet)$ based on information from the selected features $\boldsymbol{S}_n^k$. After collecting all e_n^k's, an appropriate fusion rule is required to generate a global decision regarding the system condition. Generally, there is a cost associated with each decision making depending on the true underlying hypothesis. A cost matrix $\boldsymbol{C} = \{c_{ij}\}$ is defined, where c_{ij} is the cost of decision $G = j$, while H_i is true, $i,j = 1, \ldots, N$. The optimal fusion rule is designed such that the average decision cost is minimized.

Most often, the cost of correct decisions should have minimum values. Without loss of generality, we assume that $c_{ii} = 0$, $i = 1, \ldots, N$. The expression of the average cost can be expanded as follows,

$$R = \sum_i \sum_j c_{ij} (P(H_i, G = j)$$

$$= \sum_i \sum_j c_{ij} P(H_i) \sum_{\{e_n^k\}} P(G = j, \{e_n^k\} \mid H_i) \quad (12)$$

$$\stackrel{m}{=} \sum_i \sum_j c_{ij} P(H_i) \sum_{\{e_n^k\}} P(G = j \mid \{e_n^k\}) \prod_{n=1}^{N} \prod_{k \in \Re_n} P(e_n^k \mid H_i)$$

where $\Re_n$ is the set of sensors where classifier n is designed locally. Here, c_{ij} and $P(H_i)$ are the prior knowledge and assumed known. The equality $\stackrel{m}{=}$ is due to the independence of local decision making conditioned on the underlying hypothesis. Furthermore,

$$P(e_n^k \mid H_i) = \begin{cases} (1 - P_{m,n}^k)^{e_n^k} P_{m,n}^{k\,(1-e_n^k)}, & \text{when } i = n \\ P_{f,n}^{k\,e_n^k} (1 - P_{f,n}^k)^{(1-e_n^k)}, & \text{when } i \neq n \end{cases} \quad (13)$$

where $P_{f,n}^k$ and $P_{m,n}^k$ are the probabilities of false alarm and miss, respectively, of classifier n at sensor k. They can be obtained from the classifier training processes. $P(G = j \mid \{e_n^k\})$, $j = 1, \ldots, N$ in (13) is the fusion rule that we need to design such that the average decision cost R is minimized. Equation (13) can be rewritten as follows,

$$R = \sum_{\{e_n^k\}} \sum_j P(G = j \mid \{e_n^k\}) \sum_i c_{ij} P(H_i) \prod_{n=1}^{N} \prod_{k \in \Re_n} P(e_n^k \mid H_i) \quad (14)$$

To minimize R, $P(G = j \mid \{e_n^k\}) = 1$ only if

$$j = \arg\min_j \sum_i c_{ij} P(H_i) \prod_{n=1}^{N} \prod_{k \in \Re_n} P(e_n^k \mid H_i) \quad (15)$$

This is the global decision if the data record is of a known class $\{1, \ldots, N\}$. If the data record is of an unknown class, then local classifiers tend to have 0 as their local decisions, the probability of which is $(1 - P_{f,n}^k)$, $n = 1, \ldots, N$. The obtained minimum cost of this case tends to be much larger than that of the case when the data record is of a known class. Based on this, we can modify the fusion rule as follows,

$$G = \begin{cases} j & \text{if } \min C_{j,\{e_n^k\}} \leq \lambda, j = 1, \ldots, N \\ N + 1 & \text{otherwise} \end{cases} \quad (16)$$

where $C_{j,\{e_n^k\}} = \sum_i c_{ij} P(H_i) \prod_{n=1}^{N} \prod_{k \in \Re_n} P(e_n^k \mid H_i)$, the cost of making decision j when $\{e_n^k\}$ are given, $j = 1, \ldots, N$. That is, if the minimum decision cost is above a certain threshold value, it can be determined that the system works under some unknown situations. Threshold λ can be learned from the distributions of the minimum decision cost under all the known classes by controlling the probability of misclassifying a known class as an unknown class.

6 Experimental Results

In this section, we evaluate the proposed classification framework using both synthetic and real datasets. The datasets used in the experiments are summarized in Table 1. The first four real datasets are available in [1, 24] and are widely used in the literature. Regarding the KDD99 dataset (the second column from right), the refined one in [22] is used, in which all redundant records and incorrect representations were removed. Besides, samples from unknown attacks are included for testing the unknown class detection capability of the proposed framework and they are not involved in training.

The synthetic data consists of two classes and ten features. Among the ten features, eight features are manually generated as described in Table 2.[1] Feature f_1 is a linear combination of features f_8, f_9, f_{10}. Feature f_6 is a linear combination of features f_2, f_3, f_4. 1,000 samples are randomly generated for each class.

6.1 Evaluation of The Feature Selection Method

To evaluate the proposed feature selection method, the Syn and WBCD datasets are used. For the comparison purpose, existing feature selection methods including Max-relevance and Min-redundancy (mRMR) [18], normalized mutual information feature selection (NMIFS) [6], and feature selection based on pairwise conditional mutual information (PCMI) [8] are also evaluated. In our experiment with Syn data, we first generate samples of features f_8, f_9, f_{10}. Then the sample value of feature f_1 is the average of f_8, f_9, f_{10}. The same procedure applies to feature f_6 and f_2, f_3, f_4. Other linear combinations can also be adopted. Given the values of any three features among f_1, f_8, f_9, f_{10}, the value of the rest one is known. Therefore, one of them is considered redundant if the other three are given. Similarly, one of features in f_2, f_3, f_4, f_6 is redundant if the other three are known. In addition, features f_5 and f_7 are irrelevant. This means that the final feature subset should contain six features, in which irrelevant and redundant features are not included. In our proposed method, threshold γ is set such that only six features are selected, i.e., $\gamma = 10^{-8}$. In other feature selection methods, the upper bound on the number of selected features is also set at six. Table 4 summarizes the selected features using different methods. The features are presented in the order of selection. Experimental results show that the proposed feature selection method can successfully leave out both irrelevant features $\{f_5 f_7\}$, and redundant features $\{f_6, f_{10}\}$. Meanwhile, mRMR and NMIFS methods cannot discard any redundant feature of the group . As a result, there is no room for them to select other relevant features. On the other hand, PCMI cannot

Table 1: Datasets used in experiments

datasets	Wisconsin Breast Cancer Diagnostic	Cardiotocography	Myocardial Infraction	KDD cup 1999	Synthetic
Acronyms	WBCD	CTG	MyI	KDD99	Syn
# of features	30	21	33	41	10
# of classes	2	3	5	5	2
# of samples in class 1	212	1655	74	13711 (normal)	1000
# of samples in class 2	357	295	43	9741 (DoS)	1000
# of samples in class 3		176	35	3186 (R2L)	
# of samples in class 4			98	1076 (U2R)	
# of samples in class 5			80	5106 (Probe)	
# of samples in unknown class				3750	

Table 2: Characteristics of synthetic data

	Class 1	Class 2
Feature 1 f_1	linear combination of $\{f_8, f_9, f_{10}\}$	
Feature 2 (f_2)	$\mathcal{N}(-1,1)$	$\mathcal{N}(2,1)$
Feature 3 (f_3)	$\mathcal{U}(-1,0)$	$\mathcal{U}(0,2)$
Feature 4 (f_4)	$\mathcal{N}(\alpha,1)$ $\alpha \sim \mathcal{U}(2,3)$	$\mathcal{N}(\alpha,1)$ $\alpha \sim \mathcal{U}(-2,-1)$
Feature 5 (f_5)	1	1
Feature 6 (f_6)	linear combination of $\{f_2, f_3, f_4\}$	
Feature 7 (f_7)	$\mathcal{B}(0,1,0.5)$	$\mathcal{B}(0,1,0.5)$
Feature 8 (f_8)	$\mathcal{N}(0,1)$	$\mathcal{N}(2,1)$
Feature 9 (f_9)	$\mathcal{N}(0,1)$	$\mathcal{N}(\alpha,1)$ $\alpha \sim \mathcal{U}(2,4)$
Feature 10 (f_{10})	$0.5\mathcal{N}(-2.5,1) + 0.5\mathcal{N}(2.5,1)$	$\mathcal{N}(0,1)$

[1] $\mathcal{N}(a,b)$ stands for a normal distribution with mean a and variance b. $\mathcal{U}(a,b)$ stands for a uniform distribution between a and b. $\mathcal{B}(a,b,p)$ stands for a Bernoulli distribution between two values a and b with the probability of being each value is p. $\alpha\mathcal{N}(a,b) + (1-\alpha)\mathcal{N}(c,d)$ stands for a Gaussian mixture distribution with the mixing parameter a.

Table 3: Excerpt of the synthetic data

Class	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
Class 1	-0.75	-0.46	-0.30	2.90	1.00	0.71	0.00	-0.25	0.15	-2.15
	1.07	0.83	-0.26	4.95	1.00	1.84	1.00	0.90	0.38	1.94
	-0.45	-3.26	-0.24	0.69	1.00	-0.94	1.00	-1.29	-1.42	1.36
	0.46	-0.14	-0.61	2.95	1.00	0.73	1.00	1.02	1.20	-0.83
	-0.62	-0.68	-0.57	2.31	1.00	0.35	0.00	-0.38	-0.04	-1.44
	0.37	-2.31	-0.04	2.63	1.00	0.09	1.00	-0.70	-0.70	2.52
	0.82	-1.43	-0.43	1.84	1.00	-0.01	1.00	-0.73	-0.46	3.64
	-0.36	-0.66	-0.15	2.70	1.00	0.63	1.00	-2.13	0.92	0.12
	1.53	2.58	-0.72	2.95	1.00	1.60	0.00	-0.09	0.00	4.69
	0.20	1.77	-0.38	2.60	1.00	1.33	0.00	-1.15	-0.13	1.87
Class 2	2.43	2.67	0.22	-1.01	1.00	0.63	1.00	3.89	3.84	-0.45
	0.86	1.33	1.19	0.61	1.00	1.04	0.00	1.19	2.25	-0.86
	1.38	1.60	0.86	-2.27	1.00	0.06	1.00	1.79	2.65	-0.31
	2.08	1.33	1.46	-1.79	1.00	0.33	1.00	2.08	4.38	-0.21
	2.71	2.58	0.52	-2.41	1.00	0.23	0.00	2.34	5.65	0.14
	2.48	1.22	0.19	-1.76	1.00	-0.12	0.00	2.43	5.28	-0.27
	2.83	0.94	0.90	-1.39	1.00	0.15	0.00	3.48	4.58	0.44
	1.44	2.55	1.28	-0.52	1.00	1.10	0.00	2.71	2.68	-1.09
	2.33	1.58	0.26	-1.13	1.00	0.24	1.00	1.44	5.67	-0.12
	2.78	2.36	0.91	-2.33	1.00	0.31	1.00	4.40	4.28	-0.36

Table 4: Synthetic data: selected features and classification errors

Selection method	Features	Accuracy (%)	F1 score	Computation time (s)
Proposed method	$\{f_1, f_2, f_3, f_4, f_8, f_9\}$	97.48	0.9749	521
mRMR	$\{f_6, f_1, f_4, f_2, f_9, f_3\}$	95.70	0.9576	142138
NMIFS	$\{f_6, f_4, f_1, f_3, f_2, f_9\}$	95.70	0.9576	150390
PCMI	$\{f_6, f_1, f_3, f_9, f_{10}, f_8\}$	76.48	0.7175	609

remove any redundant feature of the group $\{f_1, f_8, f_9, f_{10}\}$, while it does not select other relevant features. Our proposed method also has the highest accuracy and F1 score among all investigated feature selection methods. The processing times of mRMR and NMIFS are much higher than both the proposed method and PCMI. This is because the computational complexity of the proposed method, which is proportional to M^2d^2D, is much smaller than that of mRMR and NMIFS methods (proportional to M^3dD) form M = 2000 and d = 6.

For WBCD, the selected feature subsets and the classification errors using the four different methods mentioned above are summarized in Table 5. Empirically, with a very small threshold $\gamma = 10^{-10}$, the proposed algorithm selects six features into the final subset. Therefore, in mRMR, NMIFS and PCMI methods, the maximum number of selected features is set at six. The performance of all selected feature subsets is obtained using the 10-fold cross validation technique with a Bayesian classifier [15].

All features in Table 5 are arranged in the order of the selection. F1 score is used as an additional performance metric. The experimental results show that the proposed feature selection method has the highest accuracy and F1 score. Additionally, the proposed feature selection method consumes significantly less computational time compared with other approaches.

6.2 Multiple 1-rest Binary Classifiers Vs. One m-ary Classifier

To compare the performance of using multiple binary classifiers and using one *m*-ary classifier, CTG and MyI are used. In this experiment, two-layered neural networks are used for *m*-ary classifiers. The proposed feature selection method is applied for the design of both types of classifiers. Multiple binary decisions are combined using the minimum decision

Table 5: WBCD data: selected features and classification errors

Selection method	Features	Accuracy (%)	F1 score	Computation time (s) of feature selection algorithm
Proposed method	8, 10, 13, 29, 30, 31	85.41	0.8131	421
mRMR	28, 14, 22, 27, 4, 5	44.65	0.5693	204,088
NMIFS	28, 21, 24, 27, 2, 3	78.89	0.7297	200,102
PCMI	28, 4, 2, 22, 14, 3	62.09	0.6530	2,100

cost based fusion rule, in which $c_{ij} = 1$ if $i \neq j$ and $c_{ii} = 0$.

Tables 6 and 7 list selected features for either the m 1-rest classifiers or the one m-ary classifier for CTG and MyI data, respectively. We can see that some features are selected in the m-ary feature subset but not in any 1-rest feature subset. Some are selected in one particular 1-rest feature subset but not in the others. This is because the discriminative features are different from one kind of classifier to the others. For example, in Table 6, feature 1 is selected in Class 1 vs. rest classifier and 3-ary classifier, but not selected in Class 2 vs. rest or Class 3 vs. rest classifiers. It is because feature 1 is the baseline fetal heart rate, which is only useful to discriminate normal condition of the fetus (Class 1) from abnormal conditions (Class 2 and Class 3). Feature 9, which is the mean value of short term variability of the heart beat, is useful for 3-ary classification but does not show any discriminant characteristic of a particular class, hence, it is only selected in the 3-ary classifier. Similarly, feature 17, which is the histogram mode, is not selected in the Class 1 vs. rest classifier but is selected for all other classifiers. This is because it is only statistically useful to discriminate between the Suspect condition (Class 2) and the Pathologic condition (Class 3).

The classification accuracy of using multiple binary classifiers and single m-ary classifier is illustrated by confusion matrices in Tables 8 and 9, respectively, for CTG. Similarly, Tables 10 and 11 provide the confusion matrices for MyI. From these results, we can see that using multiple 1-rest binary classifiers can provide better classification performance than using just one m-ary classifier. This is because together with individual selected feature subsets, binary classifiers can provide better discrimination between classes.

Table 6: CTG data selected features

Classifer	Selected features	Computation time(s)
Class 1 vs. rest	1, 2, 3, 8, 10, 11, 12, 15, 16, 18, 19, 20, 21	39116
Class 2 vs. rest	2, 3, 8, 10, 11, 12, 15, 16, 17, 18, 19, 20, 21	35588
Class 3 vs. rest	2, 3, 8, 10, 11, 12, 15, 16, 17, 18, 19, 20, 21	47239
3-ary classifier	1, 2, 3, 8, 9, 10, 11, 12, 15, 16, 17, 18, 19, 20, 21	57432

Table 7: MyI data: selected features

Classifier	Selected features	Computation time (s)
Class 1 vs. rest	1, 2, 3, 4, 5, 7, 8, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33	7386
Class 2 vs. rest	1, 3, 4, 5, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33	6967
Class 3 vs. rest	1, 3, 4, 5, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33	7072
Class 4 vs. rest	1, 2, 3, 4, 5, 8, 9, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33	7412
Class 5 vs. rest	1, 2, 3, 4, 5, 7, 9, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33	7300
5-ary classifier	1, 3, 4, 5, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33	8558

Table 8: Confusion matrix of binary classifiers applied on CTG data

Predicted → Actual ↓	Class 1 (%)	Class 2 (%)	Class 3 (%)
Class 1	**96.7**	3.06	0.24
Class 2	20.88	**77.08**	2.03
Class 3	5.34	8.86	**85.8**

Table 9: Confusion matrix of 3-ary classifiers applied on CTG data

Predicted → Actual ↓	Class 1 (%)	Class 2 (%)	Class 3 (%)
Class 1	**95.44**	4.19	0.36
Class 2	28.14	**69.9**	1.97
Class 3	5.34	14.77	**79.89**

Table 10: Confusion matrix of binary classifiers applied on MyI data

Predicted → Actual ↓	Class 1 (%)	Class 2 (%)	Class 3 (%)	Class 4 (%)	Class 5 (%)
Class 1	**88.85**	3.72	0.34	2.03	5.07
Class 2	15.70	**73.26**	1.74	2.91	6.4
Class 3	12.14	0	**81.43**	3.57	2.86
Class 4	7.14	1.02	1.02	**86.99**	3.83
Class 5	4.06	0.31	0	0.63	**95**

6.3 Evaluation of The Distributed Classification Framework

To evaluate the proposed distributed classification framework, KDD99 is used in the rest of the experiments.

Table 11: Confusion matrix of 5-ary classifiers applied on MyI data

Predicted → Actual ↓	Class 1 (%)	Class 2 (%)	Class 3 (%)	Class 4 (%)	Class 5 (%)
Class 1	**76.01**	8.11	1.35	10.14	4.39
Class 2	35.47	**55.05**	0.64	5.87	2.97
Class 3	6.43	5.71	**57.14**	28.57	2.14
Class 4	15.31	8.42	10.20	**61.48**	4.59
Class 5	1.56	0.63	0.00	0.63	**97.19**

Table 12: Allocation of all features in different sensors

Sensor	Features
Sensor 1	1, 2, 3, 4, 5, 6, 7, 8, 9, 23, 24, 25, 26, 27, 28, 29, 30, 31
Sensor 2	10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22
Sensor 3	32, 33, 34, 35, 36, 37, 38, 39, 40, 41

The data in the KDD99 dataset comes from three different sources (or sensors): network sniffing data from the sniffer (Sensor 1), Sun Solaris BSM audit data from a Solaris host (Sensor 2), and disk dump data from three UNIX machines (Sensor 3), with a total 41 features. Five classes are considered: Normal, DoS, R2L, U2R and Probe. Therefore, five 1-rest classifiers are designed at each local sensor separately. A local classifier is considered as useful if it can provide higher than 50 percent accuracy (better than a random choice). In this experiment, binary classifiers 2, 3 and 5 at Sensor 2 are not useful, hence, they are eliminated. From the classifier-point-of-view, Classifier 2, Classifier 3 and Classifier 5 select only Sensor 1 and Sensor 3 as informative sources, while Classifier 1 and Classifier 4 select all three sensors. That is, $\Re_1 = \Re_4 = \{1, 2, 3\}$, and $\Re_2 = \Re_3 = \Re_5 = \{1, 3\}$.

All local binary decisions from selected sensors are sent to the fusion center. The total amount of communications required for one test data sample is twelve bits since there are a total of twelve local binary classifiers. The global decision is made according to (16). The prior distribution of the classes is generally difficult to obtain. The uniform distribution is used here to indicate the least prior knowledge, i.e., $P(H_i) = \frac{1}{5}$, $i = 1, \ldots, 5$. The cost matrix as in the KDD99 competition [5] is adopted, which is shown in Table 13. The accuracy of local decisions is represented by their local probabilities of false alarm and miss. The threshold λ used to detect unknown attacks is set based on the training dataset. Specifically, the proposed distributed detection algorithm is implemented on the validation subset which contains the samples of the five known classes. The achieved minimum decision costs of all samples are recorded and the distributions are estimated. The threshold λ is set to control the probability of misclassifying a known class as an unknown attack below 15 percent on average. We perform the experiment 10 times and report the average performance in Table 14. This table presents the confusion matrix in which the values represent the percentage of detection with respect to each of the known classes as well as a novel class. Experimental results show that the proposed detection framework can successfully detect more than 54 percent of the novel attacks and this has not been reported in any previous work. The correct detection rate of R2L is about 55 percent, which is much higher than 26.58 percent reported in previous work [10]. Similarly, the correct detection rate of U2R is higher than that of all previous methods. In addition, the majority of misclassified samples of Normal, DoS and Probe fall into the unknown class, which can be further investigated by network security experts.

Table 13: Commonly used cost matrix in the literature

Predicted → Actual ↓	Normal	DoS	R2L	U2R	Probe
Normal	0	2	2	2	1
DoS	2	0	2	2	1
R2L	4	2	0	2	2
U2R	3	2	2	0	2
Probe	1	2	2	2	0

Table 14: Detection confusion matrix of proposed method when threshold λ is used

Predicted → Actual ↓	Normal (%)	DoS (%)	R2L (%)	U2R (%)	Probe (%)	Unknown (%)
Normal	**81.88**	1.19	1.70	0	1.04	14.18
DoS	0	**82.65**	1.31	0	7.09	8.95
R2L	35.06	0	**54.62**	0	3.41	6.91
U2R	0	0	54.05	**27.03**	18.92	0
Probe	0	2.17	0.27	0	**82.82**	14.74
Unknown	5.52	1.04	29.15	0.24	9.65	**54.40**

To compare with previous classification methods applied to the KDD99 dataset presented in [10], we adopt their metrics for each cyber attack: probability of detection (PD %), which is defined as the rate of detecting an attack as any type of attacks, and false alarm rate (FAR %), which is the rate of misclassifying Normal into the concerned attack. These metrics are reasonable in that, in real networks, detecting an attack is more important than classifying it correctly to minimize its effect.

From the comparison Table 15, we see that applying to the same dataset, the proposed method has detection rates of 60.26 percent for R2L attack and 89.75 percent for U2R, which are

Table 15: Comparison results

		Probe	DoS	R2L	U2R
Proposed method	PD	98.80	97.85	60.26	89.75
	FAR	1.24	1.24	1.89	0
Layered CDF	PD	98.60	97.40	29.60	86.30
	FAR	0.91	0.07	0.35	0.05
KDD'99 winner	PD	83.30	97.10	8.40	13.20
	FAR	0.60	0.30	0.005	0.003
Multilayer Perceptron	PD	88.70	97.20	5.60	13.20
	FAR	0.40	0.30	0.01	0.05
K-Means Clustering	PD	87.60	97.30	6.40	29.80
	FAR	2.60	0.40	0.10	0.40
C4.5 (Decision trees)	PD	80.80	97.00	4.60	1.80
	FAR	0.70	0.30	0.005	0.002

much higher than previously published results. It is because the cost of misclassifying R2L or U2R attacks into Normal is higher than into other types of attacks. This helps improve the PD of these two attacks.

Note that the objective of our distributed classification framework is to minimize the average decision cost. If the cost structure changes, the classification results will also change accordingly. Hence, the proposed method is very flexible and can adapt to real life scenarios by adjusting the prior knowledge of the cost matrix and probabilities of various events.

7 Conclusions

In this paper, a new distributed classification framework is proposed, which consists of using 1-rest binary classifiers locally and a decision fusion rule combining all local decisions to produce a global decision. A new optimal feature selection method based on conditional mutual information is proposed to enhance the performance of local binary classifiers. The new feature selection method is shown to be more efficient in selecting all useful features for classification problems compared to previous methods. A decision fusion method based on the minimum mean decision cost criterion effectively combines the information from different local binary classifiers. By comparing the achieved minimum decision cost with a threshold, novel conditions of a system can be detected. Both synthetic data and various real datasets are used for evaluation. Experimental results confirm the improved performance of the proposed distributed classification framework. It is shown to be more flexible and efficient in solving classification problems.

References

[1] Acute Myocardial Infarction Data Set National Institute for Cardiovascular Outcomes Research: http://www.ucl.ac.uk/nicor/dataforresearch, 2003.

[2] R. Battiti, "Using Mutual Information for Selecting Features in Supervised Neural Net Learning," *IEEE Trans. Neural Networks*, 5(4):537-550, 1994.

[3] D. Bremner, E. Demaine, J. Erickson, J. Iacono, S. Langerman, P. Morin, and G. Toussaint, "Output-Sensitive Algorithms for Computing Nearest-Neighbor Decision Boundaries," *Discrete and Computational Geometry*, 33(4):593-604, 2005.

[4] R. O. Duda and P. E. Hart, *Pattern Analysis and Scene Classification*, Wiley, New York, 1973.

[5] C. Elkan, "Results of the KDD'99 Classifier Learning," *ACM SIGKDD Explorations*, 1:63-64, 2000.

[6] P. A. Estevez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized Mutual Information Feature Selection," *IEEE Transactions on Neural Networks*, 20(2):189-201, 2009.

[7] R. M. Fano, *Transmission of Information: A Statistical Theory of Communications*, MIT Press and John Wiley & Sons Inc., New York, 1961.

[8] F. Fleuret, "Fast Binary Feature Selection with Conditional Mutual Information," *Journal of Machine Learning Research*, 5:1531-1555, 2004.

[9] B. Guo and M. S. Nixon, "Gait Feature Subset Selection by Mutual Information," *IEEE Trans. on Systems, MAN, and Cybernetics - Part A: Systems and Humans*, 39(1):36-46, 2009.

[10] K. K. Gupta, B. Nath, and R. Kotagiri, "Layered Approach using Conditional Random Fields for Intrusion Detection," *IEEE Transactions on Dependable and Secure Computing*, 5(4):1-15, 2008.

[11] R. Gutierrez–Osuna, "Lecture Notes on Introduction to Pattern Recognition," http://courses.cs.tamu.edu/rgutier/cs790_w02/l1.pdf.

[12] N. Kwak and C. Choi, "Input Feature Selection by Mutual Information Based on Parzen Window," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(12):1667-1671, 2002.

[13] H. Liu and H. Motoda, "Feature Selection for Knowledge Discovery and Data Mining," *The Kluwer International Series in Engineering and Computer Science*, Kluwer Academic, 1998.

[14] Neural Network Toolbox, Mathworks: http://www.mathworks.com/products/neural-network/, 2012.

[15] Andrew Ng, "Machine Learning," http://www.stanford.edu/class/cs229/, Stanford, 2010.

[16] D. Parikh and T. Chen, "Data Fusion and Cost Minimization for Intrusion Detection," *IEEE Transactions on Information Forensics and Security*, 3(3):381-389, 2008.

[17] E. Parzen, "On the Estimation of a Probability Density Function and Mode," *Annals of Math. Statistics*, 2(3):1065-1076, 1962.

[18] H. Peng, F. Long, and C. Ding, "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Minredundancy," *IEEE Trans on Pattern Analysis and Machine Intelligence*, 27(8):1226-1238, 2005.

[19] G. Qu, S. Hariri, and M. Yousif, "A New Dependency and Correlation Analysis for Features," *IEEE Transactions on Knowledge and Data Engineering*, 17(9):1199-1207, 2005.

[20] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, University of Illinois Press, Urbana, IL, 1949.

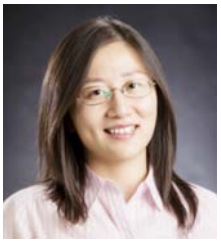
[21] Z. B. Tang, K. R. Pattipati, D. L. Kleinman, "A Distributed M-ary Hypothesis Testing Problem with Correlated Observations," *IEEE Transactions on Automatic Control*, 37(7):1042-1046, 1992.

[22] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[23] D. M. J. Tax and R. P. W. Duin, "Using Two-Class Classifiers for Multiclass Classification," *Proc. 16th International Conference on Pattern Recognition*, 2:1051-4651, 2002.

[24] UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/datasets.html, 2005.
[25] T. Wang, Y. Han, P. K. Varshney, and P. Chen, "Distributed Fault-Tolerant Classification in Wireless Sensor Networks," *IEEE Journal on Selected Areas in Communications*, 23(4):724-734, 2005.
[26] L. Xu, A. Krzyzak, and C. Suen, "Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418-435, 1992.
[27] G. P. Zhang, "Neural Networks for Classification: A Survey," *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 30(4):451-462, 2000.
[28] H. Zhang, "The Optimality of Naive Bayes," *Proc. The 17th International FLAIRS Conference* (FLAIRS2004), 1(2):3-9, 2004.
[29] X. Zhu, Y. Yuan, C. Rorres, and M. Kam, "Distributed M-ary Hypothesis Testing with Binary Local Decisions," *Information Fusion*, 5:157-167, 2004.

Hoa Dinh Nguyen received the B.S. and M.S. degrees in Electrical Engineering from Hanoi University of Science and Technology, Vietnam, and Ph.D. degree in Electrical Engineering from Oklahoma State University, Stillwater, OK, in 2000, 2002, and 2013, respectively. From 2000 to 2004, he worked as an Engineer at Vietnam Television Technology Development and Investment Company (VTC). From 2004 to 2007, he was an Expert in Posts and Telecommunications Institute of Technology (PTIT), Vietnam. Currently, he is a Lecturer in information technology at PTIT. His research interests include feature selection, classification, decision fusion, statistical signal processing, database systems and machine learning.



Qi Cheng received the B.E. degree in Electrical Engineering (highest honors) from Shanghai Jiao Tong University, Shanghai, China, in July 1999, and the M.S. and Ph.D. degrees in Electrical Engineering from Syracuse University, Syracuse, NY, in 2003 and 2006, respectively. From 1999 to 2000, she worked as a System Engineer at Guoxin Lucent Technologies Network Technologies Company, Ltd., Shanghai, China. Since August 2006, she has been with Oklahoma State University, where she is currently an Associate Professor with the School of Electrical and Computer Engineering. Her area of interest mainly focuses on statistical signal processing and data fusion with applications in distributed sensor networks. Dr. Cheng has served as Associate Editor for the IEEE Communications Letters. She is a Senior Member of the IEEE and Member of Women in Engineering ProActive Network (WEPAN).

A Rule-Based Model and Genetic Algorithm Combination for Persian Text Chunking

Samira Noferesti* and Mehrnoush Shamsfard*
Shahid Beheshti University, Tehran, IRAN

Abstract

This paper introduces a hybrid approach to Persian text chunking. The approach is that of two methods applied in sequence; a rule-based method and a genetic algorithm for shallow parsing. At the first phase, the rule-based model was applied to assign IOB tags to some tokens of an input sentence. Then, in the second phase, the search capabilities of the genetic algorithm were used to assign untagged tokens to corresponding chunks using the previously captured training information. The proposed algorithm was evaluated on the Peykareh corpus and it achieved 94.82 percent accuracy. Test results show that this proposed algorithm outperformed other algorithms for Persian text chunking.

Key Words: Rule-based model, genetic algorithm, chunking, shallow parsing, Persian language.

1 Introduction

Shallow parsing, also known as chunking or light parsing aims at discovering the main non-overlapping constituents of sentences such as noun phrases and verb phrases. It is a well-known tool used for natural language processing. It is commonly applied in many areas such as information extraction, bilingual alignment, summary generation, semantic role labeling, machine translation and named entity recognition.

Although shallow parsing has been extensively studied in other languages, investigation of the available literature revealed only two previous studies on Persian text chunking: one of which applied a rule-based algorithm of hand crafted rules [21] and the other a rule-based method to create a tagged corpus for training a neural network followed by a multilayer neural network and Fuzzy C-Means clustering, to chunk new sentences [13].

The aim of this paper was to introduce an approach for Persian text chunking using the combination of a genetic algorithm and a rule-based model. Genetic algorithms detect general chunk patterns in text and rule-based models handle special cases and exceptions to patterns in text. For the rule-based model, linguistic rules of [21] (termed general rules), were extended by analyzing errors of the genetic algorithm and defining new rules for handling these errors (termed correction rules). Although the genetic algorithm is capable of handling most of the general rules, application of these newly defined correction rules was made to speed up the work of the genetic algorithm. The proposed algorithm for text chunking contained two phases. The first phase used the rule-based model to identify some chunks. Then the second phase made good use of the search capabilities of the genetic algorithm to assign untagged tokens to corresponding chunks. Therefore, a faster genetic algorithm was achieved by producing more tagged tokens that had been generated from the rule-based model.

There is a remarkable amount of ongoing research on applying machine learning approaches to text chunking in the English language. Most machine learning approaches, such as those methods based on hidden Markov models, use information extracted from a tagged corpus to assign a suitable tag to each word according to preceding tags. Since these approaches are purely statistical, as such they are most suitable for cases that have a corpus large enough to contain all possible combinations of n-grams. In contrast, evolutionary algorithms offer a more generalized method that can be applied to any statistical model. For example, they can be applied to perform tagging operations according to the Markov model (tag prediction for a current word based on preceding tags) or improve the Markov results by using more contextual information (for example, using tags of preceding words or those of following words). In other words, HMM or other models can be used as part of the fitness function in a genetic algorithm. Therefore, a genetic algorithm provides more flexibility than any of the other classical approaches such as HMM based methods.

Accurate results have been obtained by applying evolutionary algorithms for POS tagging and stochastic tagging algorithms. According to [3] evolutionary algorithms have the advantage of being applicable to any statistical model without the need to rely on the Markov assumption (i.e., the tag given to a word is dependent only on previous words), as is the requirement in classic search algorithms for tagging, such as the widely used Viterbi algorithm [12]. Thus a hybrid approach for shallow parsing was chosen for this study. Results of the tests in this study show that our proposed algorithm outperformed other algorithms for Persian text chunking as well as the classical HMM based method.

In the following part of this paper, a brief review of the

* Electrical and Computer Engineering Department. Email: snoferesti@ece.usb.ac.ir.

related studies is discussed in Section 2. Section 3 describes the task of chunking. Section 4 introduces the annotated corpus of Persian texts. Section 5 explains our chunking model. Experiment results are discussed in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Sentence chunking was first introduced in [1]. Since then, chunking has been studied extensively. In [17] chunking algorithms were classified according to two main groups: the linguistic approach that is based on handwritten linguistic rules and learning approaches that are mostly corpus driven. Although the pioneered works were focused on rule-based methods [2], with the development of chunking corpus, most previous works applied various kinds of machine learning algorithms to the chunking task such as a support vector machine [23], conditional random field [18], transformation based learning [19], memory based learning [10, 9], decision trees [16]; the hidden Markov model [6] and hybrid methods have also been proposed [24, 7].

There are also evolutionary algorithms for the task of chunking [20, 4, 5]. In these methods, chunk classification has been seen as an optimization problem in which the best chunk tags should be assigned to the words of a sentence. As we are exploiting a genetic algorithm, the closest work to this one is that of Atkinson and Matamala [4, 5], with the difference of choosing the fitness function and involving data driven lexical information into the decisions. In fact, tests were done using the method for text chunking Persian to assess the potential of the algorithm but the method did not achieve good results. Tests showed that results improved by combining a structure based fitness function with the addition of a data driven function to calculate the probability of a specific word being inside or outside of a chunk. Tests indicated that the new proposed fitness function performed better for text in the Persian language. In addition, a genetic algorithm was combined with a rule-based model and results showed that this hybrid algorithm produced improved results.

3 Description of the Chunking Task

A chunk is a syntactic structure, which groups several consecutive words to form a phrase. The phrase structures of a sentence can be encoded using IOB [19] or IOB2 [22] styles. In the IOB tagging style, each token is tagged with one of three specific chunk tags; I (inside), O (outside), or B(begin). In the IOB2 style, which was used in this paper, the IOB tagging scheme was merged with chunk type to create a new extended tag set. For example B_NP means that the word was the first word of a noun phrase. In this work, seven chunk types were defined: noun phrase (NP), verb phrase (VP), adverb phrase (AP), adjective phrase (JP), prepositional phrase (PP), object phrase (RP) and others (O).

4 The Corpus and Tag Set

An annotated corpus of Persian text is needed in order to train and evaluate the chunker. This corpus must be annotated with POS and chunk tags. Each word in a sentence must be assigned a tag that indicates whether the word is inside or outside a chunk and shows the appropriate type of corresponding chunk.

For the current work, a subset of Persian POS tagged corpus was used, known as Peykareh[1] [8]. This collection was gathered from daily news and common texts and contained about 2.6 million manually tagged tokens. The main corpus was tagged with a rich set of POS tags consisting of 550 different tags from which 21 tags were selected for the system. Those that could be detected by the present Persian POS taggers were selected for use in the system applied to this study.

Only a portion of the Peykareh corpus containing 128,800 tokens had IOB tags. This IOB tagged dataset was created in NLP Lab-Shahid Beheshti University [14]. The chunked corpus was divided into three sets: (1) a training set including 91,933 tokens, (2) a chunked held-out data set containing 15,604 tokens and (3) a test set containing 21,263 tokens. A further held-out data set was used to define rules in the rule-based model. In fact, a portion of the Peykareh corpus was set aside that had no IOB tags and it was called the unchunked held-out data set.

5 The Proposed Algorithm

This paper proposes a hybrid approach for chunking Persian text. First, a rule-based chunker was developed to tag as many words as possible. In this stage, parts of each given sentence were assigned IOB tags. Then, a genetic algorithm was run for the tokens, which had not been assigned an IOB tag in the previous stage.

5.1 The Rule-Based Model

Shamsfard and SadrMousavi [21] presented a rule-based model for chunking Persian text. The rule-based part of this work incorporated some of the rules cited in [21]; (30 rules or less than 25 percent of rules) but in many cases the rules were changed; some rules were eliminated and some new ones (105 rules) were introduced according to the different tag sets. Some of the new rules were defined to handle errors of the genetic algorithm. Twelve files of the Peykareh corpus were used, and these included 42,006 tokens as the unchunked held-out data set. This data set was different from the chunked data corpus used for training and testing. To define the rules, the initial rule-based model and the genetic algorithm were run on the held-out data and errors were analyzed to introduce rules that would fix them. For example, in most cases a preposition was the first token of a PP chunk. Thus, a rule was identified in the initial set of rules to show that some exceptions were observed during error analysis. In Persian some compound verbs consist of prepositions or particles before or after the main verb. In fact, a preposition is part of a compound verb,

[1] This corpus also is known by its author's name, Bijankhan.

but in the Peykareh corpus it was tagged as a preposition. In such cases, the preposition must be inside the verb phrase. For example, if the previous word is the verb عبا رتند[2] (ebã:rætænd) 'include', the preposition must be inside the VP chunk.

In this way, a set of 135 hand-crafted rules was developed. Then, the most suitable sequence of rules was determined in terms of avoiding bleeding and creeping; n fact, a tree was constructed. The first level of the tree contained some general rules. Level 2 consisted of some rules for handling exceptions of the first level and so it continued. However, each node in level i handled an exception of the rule of its parent node in level i-1 (if that rule had exceptions).

At the first stage of the proposed algorithm, each rule was taken individually from the rule-set one at a time and the function was performed only if the rule was applicable to the input word.

Rules were categorized according to three classes: syntactic, morphological and lexical. Syntactic rules used part-of-speech to tag words (either as the target word or a neighboring word) in a sequence to determine the IOB tag, while morphological rules considered internal and morphological structures of a word to do this task, and the lexical rule considered real words.

Frequency counts for the rule-categories are shown in Table 1.

Table 1: Frequency counts for the rule-categories

Syntactic	morphological	Lexical
55	9	71

In the rest of this section these categories are discussed in more detail and some examples are given from each category.

5.1.1 Syntactic Rules. Some POS categories enforced a special IOB tag on words or on neighboring words. The accusative case marker را (rã:), punctuation marks, prepositions, and auxiliary verbs were among this set.

— Punctuation

At the first stage of the algorithm all punctuation marks were assigned the O tag by the following rule. However, the IOB tag of some punctuation marks such as commas could be changed during the algorithm.

If POS(X) =PUNC Then IOB_Tag(X) = O

— Accusative case marker را (rã:)

- The Persian language has an accusative case mark را (rã:) that follows the direct object. Although sometimes direct objects can appear without a marker, the marker never follows anything other than the object. For the purpose of this study, it was supposed that an accusative case marker should be put inside the chunk of the object and termed object phrase (RP). An alternative would have been to split RP into two chunks NP or AP with another chunk for the accusative case marker. We used the first alternative as it was compatible with the chunk-annotated corpus. The following rule dictated that the accusative case marker, which was shown in the corpus with the POS tag POSTP, was always inside the chunk.

If POS(X) =POSTP Then IOB_Tag(X)=I

- The word after an accusative case marker را (rã:) was determined as either the first token of the next chunk or as an outside token. In other words, if the word after the accusative case marker was not a punctuation mark with an O tag, then it would be the first token of the next chunk.

If POS(X)=POSTP And IOB_Tag(X+1)!=O
Then IOB_Tag(X+1)=B

— Prepositions

- In the Persian language the head of a prepositional phrase is always a preposition, which is followed by an NP. The following rule showed that if a current token was a preposition, then the next token would be inside the phrase.

If POS(X) =P Then IOB_Tag(X+1) =I

- In some cases a preposition was attached to the next pronoun and the whole combination was tagged as a preposition. These cases are exceptions of the above rule. For example, the word برایم (bærã:jæm) 'For me' is an abbreviation of برای من (barã:jemæn), in which برای (bærã:je) 'for' is a preposition and من (mæn) 'I'is a pronoun. Thus, the rule mentioned above was changed as follows:

If POS(X)=P and postfix(X)! =pronoun
Then IOB_Tag(X+1)=I

— Auxiliary verbs

- In a verb phrase in the Persian language, an auxiliary verb is placed before the main verb and so the main verb is inside a chunk.

If POS(X) =AUX and POS(X+1)=V
Then IOB_Tag(X+1) =I

[2]For each Persian word or phrase we write its transliteration within parenthesis and its English meaning within single quotes. International Phonetic Alphabet (IPA) is used to represent Persian language pronunciations.

For example, in the sentence به مدرسه خواهم رفت فردا من 'I will go to school tomorrow.', the auxiliary verb خواهم (xã:hæm) 'will' appeared before the main verb رفت (ræft) 'went' and together they made a verb phrase. Thus the main verb was tagged as I.

- An auxiliary verb can sometimes appear alone without a main verb after it. In such cases, an auxiliary verb was assigned as the first token of the chunk and the next token was the first token of the next chunk or outside the chunk. This rule is written as follows;

If POS(X) =AUX and POS(X+1)!=V And IOB_Tag(X+1)!=O Then IOB_Tag(X)=B And IOB_Tag(X+1) =B

For example, in the sentence باید به من قول بدهی تو 'You must promise me', the auxiliary verb باید (bã:jæd) 'must' is alone and gets the IOB tag B. In addition, the next token is tagged as B.

5.1.2. Morphological Rules. The following rules are examples of morphological rules that determine structures that take the special IOB tag.

— When a number is shown with a POS tag NUM in corpus ends in the postfix مین (mi:n), it means that it is an ordinal number. The following rule was written since the ordinal number and a proceeding noun are placed in the same chunk;

If POS(X)=NUM and postfix(X)=مین Then IOB_Tag(X+1)=I

For example, in the sentence کتاب را خواندم سومین من 'I read the third book.', the word کتاب (ketã:b) 'book' appears after the ordinal number سومین (sevomi:n) 'third' is inside the chunk.

— Another way to construct an ordinal number is by adding the letter م (mi:m)at the end of a number. This kind of ordinal number often appears after nouns. Thus, if the last character of a number is م (mi:m) and the previous token is a noun, then the number will be inside the chunk. For example, اتاق دوم (otã:gedovoum) 'second room' in a sentence dictates that the ordinal number should be inside the chunk.

If POS(X)=NUM and postfix(X)= م and POS(X-1)=NOUN Then IOB_Tag(X)=I

5.1.3 Lexical Rules. Lexical rules consider real form of words as shown in the following examples:

— Conjunctions

- Since, conjunctions like و (væ) 'and' join two phrases of the same syntactic type, if there is a و (væ) 'and' in a sentence and the previous and following words have the same POS tag, these words are in the same chunk.

If WORD(X)= و and POS(X-1)=POS(X+1) and POS(X+1)!=V Then IOB_Tag(X)=I and IOB_Tag(X+1)=I

The above rule excepts the POS tag verb. The reason is that sometimes a sentence that has only a single verb appears after the conjunction و (væ) 'and'. In these cases, the single verb is the first token of the chunk. For example, in the sentence کتاب را به او دادم و رفتم 'I gave the book to him and went.', the verb رفتم (ræftæm) 'went' is a complete sentence and it takes the IOB tag B.

- If we have *n* structures with the same POS tags separated by a comma except for the last two structures, which are separated by, the conjunction *and*, then they will be in the same chunk.

— Verbs

- To detect verb phrases, a database of compound verbs in the Persian language was used. The database of compound verbs was produced in the NLP Lab, Shahid Beheshti University and contained about 5,000 compound verbs in Persian. Whenever there was a verb in a sentence, some of the previous words were considered. If this sequence of words existed in the compound verb database, then they were tagged as a VP chunk.

— Prepositions

- If the word از (æz) 'from' appeared after a comparative adjective shown with a POS tag AJCOMP in corpus, it would be placed inside the chunk. This was shown with the following rule:

If WORD(X)= از and POS(X-1)=AJCOMP Then IOB_Tag(X)=I

After running the rule-based model, some of the tokens remain untagged. Thus, a genetic based algorithm was used to identify the remaining chunks.

5.2 Genetic Chunking Algorithm

The proposed genetic algorithm receives a natural language sentence and assigns a corresponding chunk according to previously computed training information from the annotated corpus. Formally, given a sequence of n words and corresponding POS tags, the aim is to find the most probable chunk sequence.

In our implementation, each gene can take values: I or B. Tokens, which need the IOB tag O were determined by the rule-based model. Individuals of the first generation were produced randomly. After producing an individual, all tokens of a given sentence were assigned IOB tags (some of tokens get IOB tag by the rule-based model and others get IOB tag by the genetic algorithm). The next step was to assign a type to each bounded phrase. This was done automatically by applying the following rules:

Default chunk_type =NP
If POS(first_token)=P Then chunk_type=PP
Else If POS(first_token)=ADJ Then chunk_type=JP
Else If POS(first_token)=AUX or POS(first_token)=V Then chunk_type=VP
Else If POS(first_token)=ADJ,COMP or POS(first_token) =ADJ,SUP Then chunk_type=NP
Else For each token X in the chunk
If POS(X)=ADV Then chunk_type=AP
If POS(last-token)=RA Then chunk_type=RP
For each token X in the chunk
If POS(X)= V Then chunk_type=VP

An initial population was created randomly by assigning a random IOB value to each untagged gene (some genes were assigned IOB tags from the rule-based model). These individuals were sorted according to fitness value of individuals from high to low.

Three genetic operations were used for producing the next generation.

— Selection: All individuals in the population are sorted according to fitness, so the first individual was the best fit in the generation. To perform crossover, the *i*th and *(i+1)*th individuals of the current generation were selected, where i=1,2,...,[(p+1)/2] and p was the population size. The aim of selection was to choose the fitter individuals.
— Crossover: Selected two chromosomes, crossover exchanges portioned of a pair of chromosomes at a randomly chosen point called the crossover point.
— Mutation: Selected an untagged gene randomly and toggled its value, for example if its value was *B,* it was reset to *I* and vice versa.

5.2.1 Fitness Functions. To evaluate the quality of chunks generated for an individual, four functions were used; F1, F2, F3 and F4. These functions considered the context in which a word appeared. Context consisted of a current word, one tag to the left and another to the right and the previous word.

F1 considered the sequence of POS tags for each chunk of a sentence. The required features for computing F1 are POS tags (as provided by Peykareh) of the previous word, the next word and the current word and type of current chunk. The probability of the sequence of POS tags for chunks of a sequence of n words was as follows:

$$F1 = \sum_{j=1}^{Nc} \sum_{i=1}^{L(j)-1} P(POS_{i,j^t} \mid POS_{i-1,j^t}, POS_{i+1,j^t}) \text{ t} \quad (1)$$

Where, $P(POS_{i,j^t} \mid POS_{i-1,j^t}, POS_{i+1,j^t})$ represents the probability that given the current chunk j with type t, the POS_i tag occurs when the previous word in the j-th chunk had the POS_{i-1} tag and the next word in the j-th chunk had the POS_{i+1} tag. Nc was the number of chunks in a sentence and $L(j)$ was the length of chunk j.

F2 function computed the probability of taking IOB tag x if the current word was $word_i$ and the previous word was $word_{i-1}$.

$$F2 = \sum_{i=1}^{n} P(IOB\_Tag_x \mid word_i, word_{i-1}) \quad (2)$$

Where, n was the number of tokens in an input sentence. In order to compute F1 and F2 functions, the HMM model can be used with the Viterbi algorithm.

For computing F3 function, a data driven approach was applied to calculate the probability that a specific word has a special IOB tag.

$$F3 = \sum_{i=1}^{n} P(IOB\_Tag_x \mid word_i) \quad (3)$$

Where, $P(IOB\_Tag_x \mid Word_i)$ represents the probability of i-th token in the sentence has IOB_Tag$_x$.

F3 is defined because some words in Persian were mostly assigned a special IOB tag. For example, the word شاید (shã:jæd) 'maybe' appeared as the first token of a chunk in most cases.

F4 function was the probability that a token gets IOB tag x when it occurs after a specific word in the training corpus.

$$F4 = \sum_{i=1}^{n} P(IOB\_Tag_x \text{ for } token_i \mid word_{i-1}) \quad (4)$$

This function was defined because some words in Persian enforced special IOB tags for the next token. For example, the word به استثناء (be estesnã:e) 'except' may be enforced such that the next token is placed inside the chunk.

The effect of each of these functions is assessed in Section 6. The following fitness function was used to evaluate the genetic algorithm:

$$Fitness\ Function = \sum_{i=1}^{4} w_i \cdot F_i \quad (5)$$

Where $w_i s$ are constant parameters chosen from [0,1] and show relative importance of syntactic and lexical information. It was assumed that 0 is a legal value to show the effect of removing one or more functions from the formula. To adjust w_i parameters in the fitness function formula, variable

structure-learning automata were applied on chunked held-out data. For more information you can see [15]. Finally, the value of w_1 was set to 0.3 and the values of w_2, w_3 and w_4 were set to 1.

6 Experiments

To evaluate the performance of our proposed algorithm, three measures were taken [25]: accuracy (the percentage of correctly predicted output classes containing all types of chunks included O-class), precision (the percentage of predicted chunks that were correct) and recall (the percentage of predictable chunks that were found). A chunk was only counted as correct when its boundaries, its type and its function were all identified correctly.

Since performance was related to both precision and recall, the *F*-measure was given as the final evaluation.

$$F_{measure} = \frac{2.precision.recall}{precision + recall} \quad (6)$$

6.1 Tuning Parameters of the Genetic Algorithm

The efficiency of a genetic algorithm greatly depends on how its parameters are tuned. To adjust the genetic parameters selected, a set of 15,604 tokens were selected, as the chunked held-out data set. Then, the proposed algorithm was run on this set.

Beginning with a baseline configuration, such as Dejong's setting [11] with 1,000 generations, 50 chromosomes in each generation and 0.6 for crossover probability, the algorithm was run for different mutation probabilities (P_m) from 0.01 to 0.3. Figure 1 shows that the best results were obtained using the mutation probability 0.01.

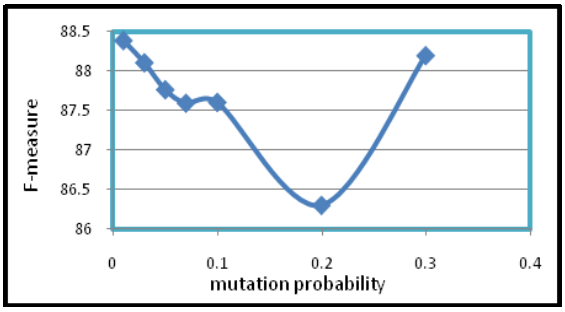


Figure 1: Average fitness values of executions of the GA using different mutation probabilities

In the same way, crossover probability was set to 0.4. In Figure 2 the results of running the genetic algorithm using mutation probability 0.01, crossover probability 0.4, population size 50 and different number of generations are shown.

Table 2 shows the optimum values of genetic algorithm parameters.

A further experiment was done to assess the effect of each part of the fitness function on chunker performance. As mentioned before, the fitness function consisted of four functions. Function F1 considered syntactic information and others considered lexical information. To assess the effect of these functions, lexical functions were removed one by one and the proposed algorithm was run with the new fitness function. Table 3 shows the effect of different fitness functions on performance of the proposed algorithm. The first column presents the fitness function. The second column shows performance of the proposed hybrid algorithm by using this fitness function in terms of F-measure. Finally, the last column shows accuracy of the genetic algorithm by using this fitness function.

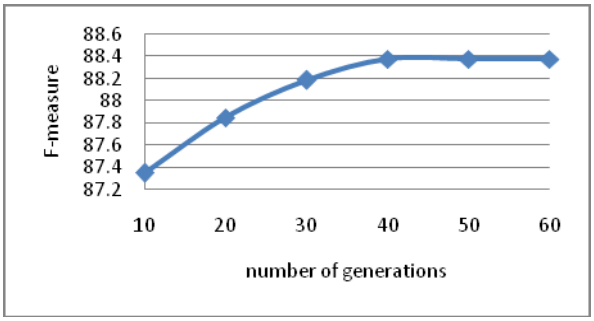


Figure 2: Average fitness values of executions of the GA using different number of generations and population sizes

Table 2: Optimum values of the genetic algorithm parameters

Genetic parameter	Value
Number of generations	40
Population size	50
Mutation probability	0.01
Crossover probability	0.4

Table 3: The effect of different fitness functions on the performance of the proposed algorithm

Fitness function	Proposed algorithm F-measure	Genetic algorithm accuracy
$w1.F1 + w2.F2 + w3.F3 + w4.F4$	88.06	87.60
$w1.F1 + w3.F3 + w4.F4$	86.70	86.01
$w1.F1 + w2.F2 + w4.F4$	84.71	83.21
$w1.F1 + w2.F2 + w3.F3$	86.27	85.01

6.2 Effectiveness of the Proposed Algorithm

The experiment applied 91,933 chunk-annotated tokens as the training set and 21,263 tokens as the test set. Parameter settings shown in Table 2 were used for the genetic algorithm.

Table 4 compares overall accuracy from the combination of the rule-based model and the genetic algorithm. Approximately 82 percent of tokens were tagged by the rule-based model with 96.39 percent accuracy. In fact, from tokens in the test set, 17,472 tokens were tagged by the rule-based model and among them 16,841 tokens were assigned correct

tags. In contrast, the genetic algorithm assigned correct tags to 3,321 tokens from 3,791 tokens and achieved 87.60 percent accuracy.

Table 4: Comparing the accuracy of the rule-based model versus genetic algorithm

	#tagged tokens	#correctly tagged tokens	Accuracy
Rule based model	17472	16841	96.39
Genetic algorithm	3791	3321	87.60

In the next experiment, performance of the proposed genetic algorithm was compared to that of the HMM model (see Table 5). For this reason, the HMM model was used with big rams and Viterbi decoding. At first, the rule-based model was run. After that, the HMM model tagged tokens, which didn't get tagged by the rule-based model. In fact, the aim was to examine the effect of using the genetic algorithm instead of the Viterbi algorithm and results demonstrated that the proposed algorithm outperformed the Viterbi algorithm. Simulation results showed that the heuristic nature of the genetic algorithm was useful for tagging.

Table 5: Comparing the proposed algorithm with HMM model

	Accuracy	Precision	Recall	F_measure
Proposed chunker	94.82	87.70	88.43	88.06
HMM model	93.56	86.97	85.74	86.35

Table 6 compares the approaches applied to these tests with other available chunkers for Persian text. As can be seen, our proposed algorithm out performed the other algorithms. The recall was computed (see Table 5) but was not compared to others because they reported only precision.

Table 6: Comparing the performance of the proposed algorithm with other methods

Method	Precision
Shamsfard and SadrMousavi, 2008 [21]	70%
Kiani et al., 2009 [13]	85.7%
Our proposed method	87.70%

In the above experiments correct POS tags were used from the Peykareh corpus. The reason for this was that results from the proposed algorithm were compared to those from other available chunkers and comparison showed that the POS tags were correct. In order to examine the effect of using estimated tags, the test data was tagged by using Tnt tagger with 96.73 percent accuracy. In this experiment, the proposed algorithm achieved the F-measure of 87.46 percent.

Since about 82 percent of tokens got IOB tags in the first stage and the training corpus was analyzed only once (at the first iteration), the genetic algorithm found the solution in reasonable time.

7 Conclusion and Future Work

This paper proposes an approach for text chunking Persian that combines genetic algorithms with rule-based models. Genetic algorithms provide a search strategy to learn general chunk patterns in text optimizing a measure of probability that is effective globally. However, the rule-based model handles special cases and exceptions to general patterns. Results of the tests reported in this study show that the proposed algorithm outperformed other algorithms for Persian text chunking and classical HMM based methods.

Although this paper presents an algorithm for Persian text chunking, the principles can be applied to other languages. A genetic algorithm can be used for any language to find common statistical patterns for chunking. Obviously, there may be exceptions to these patterns. So some rules were defined to handle exceptions in the rule-based model that served to improve performance of the genetic algorithm. In fact, hybridizing a genetic algorithm with the rule-based model improves performance of the chunking process.

In future work, linguistic rules may be extended by analyzing errors of test data. It is also observed that input of the IOB-tag set has a major influence on accuracy. Errors in the training data have caused some problems, and these can be reduced by correcting the training data.

In addition, it is intended that new attributes be added to the fitness function of the genetic algorithm. One advantage of the genetic algorithm compared to other classical approaches such as HMM based methods is that new attributes can be added to the system and this facilitates examination of the effect of different attributes on tagging without altering the system's basic structure. Thus, tests will be done on new attributes applied to the fitness function of the genetic algorithm and to evaluate effects on chunking accuracy.

References

[1] S. Abney, "Parsing by Chunks," *Principle-Based Parsing*, R. Berwick, S. Abney and C. Tenny, Ed., Kluwer Academic, 1991.

[2] S. Abney, "Partial Parsing via Finite-State Cascades," *Natural Language Engineering*, 2(4):337-344, 1996.

[3] E. Alba, G. Luque, and L. Araujo, "Natural Language Tagging with Genetic Algorithms," *Information Processing Letters*, 100:173-182, 2006.

[4] J. Atkinson and J. Matamala, "Evolutionary Shallow Parsing," *Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*, pp 420-425, 2009.

[5] J. Atkinson and J. Matamala, "Evolutionary Shallow Natural Language Parsing," *Computational Intelligence*, 28(2):156-175, 2012.

[6] S. Baskaran, "Hindi POS Tagging and Chunking," *Proceedings of the NLPAI Machine Learning 2006 Competition*, 2006.

[7] R. A. Bhat and D. M. Sharma, "A Hybrid Approach to Kashmiri Shallow Parsing," The 5th Language and

Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, 2011.
[8] M. BijanKhan, "The Role of the Corpus in Writing a Grammar: an Introduction to a Software," *Iranian Journal of Linguistics*, 19(2), 2004.
[9] A. V. Bosch and S. Buchholz, "Shallow Parsing on the Basis of Words Only: A Case Study," *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, pp 433-440, 2002.
[10] W. Daelemans, S. Buchholz, and J. Veestra, "Memory Based Shallow Parsing," *Journal of Machine Learning Research Archive*, 2:559-594, 2002.
[11] K. A. DeJong and W. M. Spears, "An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms," *Proceedings of the First Workshop Parallel Problem Solving from Nature*, Springer-Verlag, Berlin, pp 38-47, 1990.
[12] G. D. Forney, "The Viterbi Algorithm," *Proceedings of the IEEE*, 61(3):268-278. 1973.
[13] S. Kiani, T. Akhavan, and M. Shamsfard, "Developing A Persian Chunker using a Hybrid Approach," *Proceedings of the International Multiconference on Computer Science anid Information Technology*, pp. 227-233, 2009.
[14] S. Noferesti, "Building IOB Tagged Dataset", Unpublished Results, Technical Report, NLP-Lab., Shahid Beheshti University, 2010.
[15] S. Noferesti and M. Rajayi "A Hybrid Algorithm Based on Ant Colony System and Learning Automata for Solving Steiner Tree Problem," *International Journal of Applied Mathematics and Statistics*, 22(11):79-88, 2011.
[16] S. C. Pammi and K. Prahallad, "POS Tagging and Chunking using Decision Forests," Workshop on Shallow Parsing for South Asian Languages, 2007.
[17] F. Pla, A. Molina, and N. Prieto, "An Integrated Statistical Model for Tagging and Chunking Unrestricted Text," *Proceedings of the Third International Workshop on Text, Speech and Dialog*, pp. 15-20, 2000.
[18] Q. Qi, and L. Liu, "Chinese Chunking Based on Frequency used Words," *Computer Engineering and Technology (ICCET)*, 2:432-435 (2010).
[19] L. A. Ramshaw and M. P. Marcus, "Text Chunking using Transformation-Based Learning," *Proceedings of the Third Workshop on Very Large Corpora*, pp 82-94, 1995.
[20] J. I. Serrano and L. Araojo, "Evoluntionary Algorithm for Noun Phrase Detection in Natural Language Processing," *Proceedings of the 2005 IEEE Congress on Evolutionary Computing*, pp. 640-647, 2005.
[21] M. Shamsfard. and M. SadrMousavi, "Thematic Role Extraction using Shallow Parsing," *International Journal of Computational Intelligence*, 4(2):126-132, 2008.
[22] K. S. Tjong and S. Buchholz, "Introduction to the CoNLL-2000 Shared Task: Chunking," *Proceedings of Conference on Natural Language Learning*, pp 127-132, 2000.
[23] Tsuchiya, Shime and Takagi, "Chunking Japanese Compound Functional Expressions by Machine Learning," 11th Conference of the European Chapter of the Association for Computational Linguistics, *Proceedings of the Workshop on Multi-Word-Expressions in a Multilingual Context*, pp. 25-32, 2006.
[24] A. Wajid and H. Samad, "A Hybrid Approach to Urdu Verb Phrase Chunking," *Processing of 8th Workshop on Asian Language Resources* (*ALR-8*), pp 137-143, 2008.
[25] Y. Ch. Wu, Ch. H. Chang, and Y. Sh Lee, "A General and Multi-Lingual Phrase Chunking Model Based on Masking Method," *Proceedings of the 7th International Conference on Computational Linguistics and Intelligent Text Processing*, Mexico City, Mexico, LNCS 3878:144-155, 2006.

Samira Noferesti is a Ph.D student at ShahidBeheshti University. Her interests include artificial intelligence, natural language processing and opinion mining.

Mehrnoush Shamsfard received her BS and MS both in Computer Software Engineering from Sharif University of Technology, Tehran, Iran. She received her PhD in Computer Engineering-Artificial Intelligence from AmirKabir University of Technology in 2003.

Dr. Shamsfard is an Assistant Professor at Shahid Beheshti University from 2004. She is the head of NLP Research Laboratory of Electrical and Computer Engineering Faculty. Her main fields of interest are natural language processing, ontology engineering, text mining and semantic web.

Instructions For Authors

The International Journal of Computers and Their Applications is published multiple times a year with the purpose of providing a forum for state-of-the-art developments and research in the theory and design of computers, as well as current innovative activities in the applications of computers. In contrast to other journals, this journal focuses on emerging computer technologies with emphasis on the applicability to real world problems. Current areas of particular interest include, but are not limited to: architecture, networks, intelligent systems, parallel and distributed computing, software and information engineering, and computer applications (e.g., engineering, medicine, business, education, etc.). All papers are subject to peer review before selection.

A. Procedure for Submission of a Technical Paper for Consideration

1. Email your manuscript to the Editor-in-Chief, Dr. Qiang Zhu, qzhu@umich.edu.

2. Illustrations should be high quality (originals unnecessary).

3. Enclose a separate page (or include in the email message) the preferred author and address for correspondence. Also, please include email, telephone, and fax information should further contact be needed.

B. Manuscript Style:

1. The text should be **double-spaced** (12 point or larger), **single column** and **single-sided** on 8.5 X 11 inch pages.
2. An informative abstract of 100-250 words should be provided.
3. At least 5 keywords following the abstract describing the paper topics.
4. References (alphabetized by first author) should appear at the end of the paper, as follows: author(s), first initials followed by last name, title in quotation marks, periodical, volume, inclusive page numbers, month and year.
5. Figures should be captioned and referenced.

C. Submission of Accepted Manuscripts

1. The final complete paper (with abstract, figures, tables, and keywords) satisfying Section B above in **MS Word format** should be submitted to the Editor-in-Chief.
2. The submission may be on a CD/DVD or as an email attachment(s) . **The following electronic files should be included:**

    - Paper text (required).
    - Bios (required for each author).
    - Author Photos (jpeg files are required by the printer).
    - Figures, Tables, Illustrations. These may be integrated into the paper text file or provided separately (jpeg, MS Word, PowerPoint, eps).

3. Specify on the CD/DVD label or in the email the word processor and version used, along with the title of the paper.

4. Authors are asked to sign an ISCA copyright form (http://www.isca-hq.org/j-copyright.htm), indicating that they are transferring the copyright to ISCA or declaring the work to be government-sponsored work in the public domain. Also, letters of permission for inclusion of non-original materials are required.

Publication Charges

After a manuscript has been accepted for publication, the contact author will be invoiced for publication charges of **$50.00 USD** per page (in the final IJCA two-column format) to cover part of the cost of publication. For ISCA members, $100 of publication charges will be waived if requested..

January 2014