

M.F. ULUAT

METU

2014

**A FUZZY LOGIC BASED
ENSEMBLE ADAPTIVE TILE PREFETCHING**

MEHMET FATİH ULUAT

AUGUST 2014

A FUZZY LOGIC BASED ENSEMBLE ADAPTIVE TILE PREFETCHING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET FATİH ULUAT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

AUGUST 2014

Approval of the thesis:

A FUZZY LOGIC BASED ENSEMBLE ADAPTIVE TILE PREFETCHING

submitted by **MEHMET FATİH ULUAT** in partial fulfillment of the requirements
for the degree of **Doctor of Philosophy in Computer Engineering Department,**
Middle East Technical University by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering**

Prof. Dr. Veysi İşler
Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Assoc. Prof. Dr. Halit Oğuztüzün
Computer Engineering Dept., METU

Prof. Dr. Veysi İşler
Computer Engineering Dept., METU

Assoc. Prof. Dr. Tolga Can
Computer Engineering Dept., METU

Assoc. Prof. Dr. Tolga Çapın
Computer Engineering Dept., TED University

Asst. Prof. Dr. Hacer Yalın Keleş
Computer Engineering Dept., Ankara University

Date:

28/08/2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: MEHMET FATİH ULUAT

Signature :

ABSTRACT

A FUZZY LOGIC BASED ENSEMBLE ADAPTIVE TILE PREFETCHING

Uluat, Mehmet Fatih
Ph.D., Department of Computer Engineering
Supervisor: Prof. Dr. Veysi İşler

August 2014, 142 pages

Prefetching is a process in which necessary portion of data is predicted and loaded into memory beforehand. The increasing usage of geographic data in different types of applications motivated the development of different prefetching techniques. These techniques are usually developed for specific type of applications such as 2D geographic information systems or 3D visualization applications and crafted for corresponding navigation patterns. However, as boundary between these application types blurs, these techniques become insufficient for hybrid application types such as digital moving maps. This type of applications possess capabilities from both of these domains and exhibit various navigation patterns. Therefore, a group of prefetching techniques should be used together to handle different requirements and navigation patterns. In this study, a priority based tile prefetching approach is proposed which enables ensemble usage of different prefetching techniques at the same time. The proposed approach manages these techniques dynamically through a fuzzy logic based inference engine to increase prefetching performance and to adapt to various behaviors exhibited. This engine performs adaptive decisions about contribution of each technique according to their individual performance and activity level. The results obtained from experiments showed that up to 25% increase in prefetching performance is achieved with proposed adaptive ensemble usage over single technique usage. A generic model

for prefetching techniques is also developed and used to describe given approach. Finally, a cross-platform software framework with five different prefetching techniques are developed to let other users utilize proposed approach.

Keywords: Prefetching, GIS, fuzzy logic, adaptive prefetching, tile prefetching, raster data.

ÖZ

COĞRAFI VERİLER İÇİN BULANIK MANTIK İLE ÖN YÜKLEME YAKLAŞIMI

Uluat, Mehmet Fatih
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Veysi İşler

Ağustos 2014, 142 sayfa

Uygulamalar için gerekli olan verilerin önceden tahmin edilerek belleğe alınması işlemine ön yükleme denilmektedir. Uygulamalar tarafından coğrafi veri kullanımının artması ile birçok ön yükleme tekniği geliştirilmiştir. Bu tekniklerin birçoğu iki boyutlu coğrafi bilgi sistemleri veya üç boyutlu görselleştirme uygulamaları ve bunlara özgü navigasyon türleri için özelleşmiştir. Bu yaklaşımlar, belirli bir tip uygulama türü için ön yakalama ihtiyaçlarını karşılasalar da, kayan harita yazılımları gibi bünyesinde farklı kabiliyet ve navigasyon türlerini içeren uygulamalar için yetersiz kalmaktadır. Bu sebeple, tek bir ön yükleme tekniği kullanmak yerine, birden fazla teknik aynı anda kullanarak, farklı kabiliyet ve navigasyon türlerinin ihtiyaçları karşılanmalıdır. Bu çalışmada, çeşitli ön yükleme tekniklerinin birlikte kullanılmasına olanak sağlayan öncelik tabanlı bir pafta ön yükleme yaklaşımı sunulmuştur. Sunulan yaklaşım ile farklı ön yükleme tekniklerinin etkinlikleri, dinamik olarak bulanık mantık kullanılarak geliştirilmiş olan bir çıkarım motoru ile kontrol edilmektedir. Bu motor sayesinde, uygulama tarafından sergilenebilecek farklı kabiliyet ve navigasyon şekillerine uyum sağlanabilmektedir. Gerçekleştirilen deneyler ile bu yöntemin farklı kabiliyetleri bünyesinde taşıyan uygulamalarda, birlikte kullanımda tekil kullanıma göre 25%'e yaklaşan bir ön yükleme performans artışı sağlandığı gösterilmiştir. Bunun ile birlikte diğer ön yükleme yaklaşımlarını ifade etmek için kullanılabilecek genel bir

ön yükleme modeli geliştirilmiştir. Sunulan yaklaşım bu model üzerinden açıklanmıştır. Gerek farklı uygulama türlerini temsil etmek, gerekse deneylerde kullanılmak üzere beş farklı ön yükleme tekniği tasarlanmıştır. Bu yaklaşım geliştirilen bir yazılım alt yapısı ile diğer kullanıcıların kullanımına sunulmuştur.

Anahtar Kelimeler: Ön belleğe alma, CBS, bulanık mantık, uyarlanabilir ön belleğe alma, coğrafik veri.

To My Daughter and Beloved Wife

ACKNOWLEDGMENTS

The author wishes to gratefully thank his supervisor Prof. Dr. Veysi İşler for his invaluable guidance, advice and encouragements for this research.

The author would also like to thank Assoc. Prof. Dr. Tolga Can and Assoc. Prof. Dr. Tolga Çapın for their suggestions and comments.

Finally, the author wishes his special thanks to his family for their patience, support and motivation.

TABLE OF CONTENTS

ABSTRACT.....	v
ÖZ	vii
ACKNOWLEDGMENTS	x
TABLE OF CONTENTS.....	xi
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
LIST OF ABBREVIATIONS.....	xviii
CHAPTERS	
1. INTRODUCTION	1
1.1 Problem.....	3
1.2 Contributions	7
1.3 Thesis Overview	8
2. PREFETCHING	9
2.1 Studies from GIS Domain	13
2.2 Studies from Visualization Domain.....	22
3. FUZZY LOGIC	31
3.1 Fuzzification Step	35
3.2 Fuzzy Processing Step	37
3.3 Defuzzification Step	39
3.4 An Application Example of Fuzzy Logic	41
4. ENSEMBLE ADAPTIVE PREFETCHING	45
4.1 EAP Architecture.....	45
4.2 Criterion Term	49
4.3 Forms of Data	52
4.4 Prefetching Steps and Prioritization	53
4.5 Fuzzy Logic-Based Adaptive Weight Balancer	67
4.6 Ensemble Adaptive Prefetching Approach and Usage Scenario ...	79

5. DEVELOPED SET OF CRITERIA.....	91
5.1 2D MapView Criterion	91
5.2 Retrospective Adaptive Prefetching (RAP) Criterion.....	93
5.3 2D/3D Distance Criterion	93
5.4 3D Field of View (FOV) Criterion	94
5.5 Point of Interest (POI) Criterion	95
5.6 Vertical Cross Section (VCS) Analysis Criterion.....	96
6. EXPERIMENTATION AND DISCUSSION	99
6.1 Measurement Parameters	99
6.2 Experimentation One	100
6.3 Experimentation Two.....	107
6.4 Experimentation Three.....	109
7. CONCLUSION AND FUTURE STUDY	115
REFERENCES.....	117
APPENDICES	
A. EAP FRAMEWORK	123
A.1 EAP Packages	123
A.2 EAP Classes	125
A.3 EAP Utilization.....	132
B. AN EXAMPLE CONFIGURATION FILE	135
CURRICULUM VITAE	141

LIST OF TABLES

TABLES

Table 2.1 Characteristics of web objects [29]	14
Table 2.2 Neighbor tiles to prefetch according to Easting and Southing	19
Table 2.3 Execution of grading formula for sample scenario with history depth 5	20
Table 3.1 Example accumulation methods.....	39
Table 4.1 WB-FIE fuzzy logic rule matrix (rule base)	72

LIST OF FIGURES

FIGURES

Figure 1-1 Data storage levels with corresponding speed and capacities	1
Figure 1-2 The raster data tiling and layering	4
Figure 2-1 A simple usage of prefetching	9
Figure 2-2 Cooperation of server and client-side entities for prefetching [23]	10
Figure 2-3 Google Maps tiling schema [26].....	11
Figure 2-4 The first three steps of Hilbert Curve [5].....	16
Figure 2-5 Direction vector on a 2-dimensional space [28]	17
Figure 2-6 Sample navigation scenario when history depth is 5	19
Figure 2-7 System architecture of [31].....	20
Figure 2-8 Abstracted architecture of Web GIS systems [8].....	21
Figure 2-9 System pipeline of [7].....	25
Figure 2-10 Scene Graph Representation [20]	26
Figure 2-11 The out-of-core rendering approach of the iWalk system [21]	27
Figure 2-12 A section of model that colored according to PLP [21]	28
Figure 3-1 Traditional logic representation of a discrete temperature value.....	31
Figure 3-2 Fuzzy logic representation of temperature values	32
Figure 3-3 Fuzzy logic graph that illustrates the clothing choice according to temperature	33
Figure 3-4 Fuzzy logic system process algorithm	34
Figure 3-5 An overview of a fuzzy logic system	34
Figure 3-6 Membership function chart.....	36
Figure 3-7 Different membership function shapes. S, Z, \wedge , and π shapes are illustrated from left to right [39].....	36
Figure 3-8 Example temperature fuzzy set with five \wedge shaped membership functions	37
Figure 3-9 Example rule set for clothing decision	37

Figure 3-10 Rules with multiple input conditions that are logically linked with AND and OR relationships	38
Figure 3-11 A fuzzy set with multiple rules.....	38
Figure 3-12 An example centroid calculation of an output membership function	40
Figure 3-13 The input membership functions	41
Figure 3-14 The rule base.....	42
Figure 3-15 The output membership functions	42
Figure 3-16 The input value and corresponding activated membership functions	43
Figure 3-17 The corresponding fuzzy output values.....	43
Figure 3-18 The individual fuzzy output curves	44
Figure 3-19 The aggregated fuzzy output curve	44
Figure 3-20 The calculated centroid value	44
Figure 4-1 The EAP architecture	46
Figure 4-2 The forms of data.....	52
Figure 4-3 Prefetching steps and forms of data through this process	54
Figure 4-4 Raster Metadata index file structure	55
Figure 4-5 a) The tile numbering, b) load-order for each direction in RAP according to (a).....	57
Figure 4-6 2D Navigation prioritization policy.....	60
Figure 4-7 3D FOV prioritization policy	61
Figure 4-8 How outside tiles are being prioritized in 3D FOV policy.....	62
Figure 4-9 Analysis prioritization policy	63
Figure 4-10 The architecture of a weight balancer-fuzzy inference system	68
Figure 4-11 The criterion activity level membership functions	70
Figure 4-12 The criterion correction level membership functions.....	70
Figure 4-13 WB-FIE fuzzy logic rule list (rule base)	71
Figure 4-14 The mapping of list based rule to matrix representation	72
Figure 4-15 The criterion weight operation membership function	73
Figure 4-16 The input values and CCL/CAL membership functions	74
Figure 4-17 The four triggered rules are shown in circles	75

Figure 4-18 The operations occurred at fuzzy processing step for given triggered rule with corresponding mappings	75
Figure 4-19 The selected output for triggered rule.....	76
Figure 4-20 The second triggered rule with corresponding mappings	76
Figure 4-21 The third triggered rule with corresponding mappings	77
Figure 4-22 The fourth triggered rule with corresponding mappings	77
Figure 4-23 The outputs that are obtained from triggered rules.....	78
Figure 4-24 The accumulated output of fuzzy processing step and application of centroid method for defuzzification	78
Figure 4-25 The output of WB-FIE to action mapping	79
Figure 4-26 Overview of EAP usage.....	80
Figure 4-27 Example digital moving map application 2D map displays [46, 47].	85
Figure 4-28 Example digital moving map application 2D map displays (cont.) [48]	85
Figure 4-29 Example digital moving map application 3D map displays [46, 48].	86
Figure 4-30 Example digital moving map application VCS analysis display [48]	86
Figure 4-31 Tiles that will be prefetched by 2D map view criterion	87
Figure 5-1 Overview of 2D Map View criterion.....	92
Figure 5-2 a) The fixed order of tiles that will be prefetched in east and south west direction [4], b) the dynamic order in EAP that determined according to navigator location.	93
Figure 5-3 Overview of 2D Distance criterion and pivot point.....	94
Figure 5-4 Overview of FOV Criterion and its parameters.....	95
Figure 5-5 Point of Interest criterion usage for radar POI.....	96
Figure 5-6 Overview of VCS analysis criterion	97
Figure 6-1 The designed flight trajectory that scenarios are executed through with corresponding behaviors exhibited through this path.....	102
Figure 6-2 The overall hit ratios (HR) obtained from executed scenarios	103
Figure 6-3 The CCL values that are obtained from adaptive EAP scenario execution.....	104

Figure 6-4 The CAL values that are obtained from adaptive EAP scenario execution	104
Figure 6-5 The criteria weight values that are controlled and changed by adaptive WB-FIS	105
Figure 6-6 The number of direct (blue) and prefetch requests (red) initiated	105
Figure 6-7 The designed flight trajectory for experiment 2	107
Figure 6-8 The overall HRs obtained from execution of experiment 2 scenarios	108
Figure 6-9 The designed first route for experiment 3	110
Figure 6-10 The designed second route for experiment 3	110
Figure 6-11 The designed third route for experiment 3	111
Figure 6-12 The first route executions and corresponding overall HRs	112
Figure 6-13 The second route executions and corresponding overall HRs	113
Figure 6-14 The third route executions and corresponding overall HRs	114
Figure A-1 Overview of VCS analysis criterion	124
Figure A-2 Overview of EAP core package class diagram	126
Figure A-3 Overview of EAP criteria package class diagram	130
Figure A-4 Overview of EAP loaders package class diagram	131
Figure A-5 Overview of EAP navigator package class diagram	132

LIST OF ABBREVIATIONS

Abbreviation or Symbol	Text
2D.....	2 Dimension
3D.....	3 Dimension
API.....	Application Programming Interface
CAD.....	Computer Aided Design
CAL.....	Criterion Activity Level
CCL.....	Criterion Correction Level
CPR.....	Correct Prediction Ratio
CPU.....	Central Processing Unit
CWO.....	Criterion Weight Operation
DTED.....	Digital Terrain Elevation Data
EAP.....	Ensemble Adaptive Prefetching
EVF.....	Extended View Frustum
FI.....	Fuzzy Input
FO.....	Fuzzy Output
FLS.....	Fuzzy Logic System
FOV.....	Field of View
GIS.....	Geographic Information Science
GPS.....	Global Positioning System
GPU.....	Graphical Processing Unit
HCBP.....	Hilbert Curve Based Prefetching
HR.....	Hit Ratio
HTML.....	Hyper Text Markup Language
I/O.....	Input / Output
ID.....	Identifier
INS.....	Inertial Navigation Systems
LFU.....	Least Frequently Used

LOD.....	Level of Detail
LRU.....	Least Recently Used
MRU.....	Most Recently Used
NDK.....	Native Development Kit
NSMC.....	Neighbor Selection Markov chain
PKM.....	Previous-K-Movements
PLP.....	Prioritized Layered Projection
POI.....	Point of Interest
PSS.....	Pyramidal Selection Scheme
RAP.....	Retrospective Adaptive Prefetching
SLD.....	Spatial Locality by Distance
TDM.....	Textured Depth Mesh
VCS.....	Vertical Cross Section
WB-FIE.....	Weight Balancer Fuzzy Inference Engine
XML	Extended Markup Language

CHAPTER 1

INTRODUCTION

Recent advances in computer and acquisition technology increase the amount and availability of geographic data used by applications. This data usually gets through different data storage levels where each level has different speeds and characteristics. How speed and storage change through each level is illustrated in Figure 1-1. At the bottom level, data is stored in data servers which have very big but slow storage devices. On top of that level, local hard disks are being used as storage medium which are usually faster than data servers but have less storage capacities. Then application loads this data into main memory which is also known as Random Access Memory (RAM). RAM provides better read and write speeds than hard disks where reading/writing operations depend on physical locations of storage medium and are very slow. Finally, cache memory is smaller than main memory and accessed by CPU quicker than regular RAM.

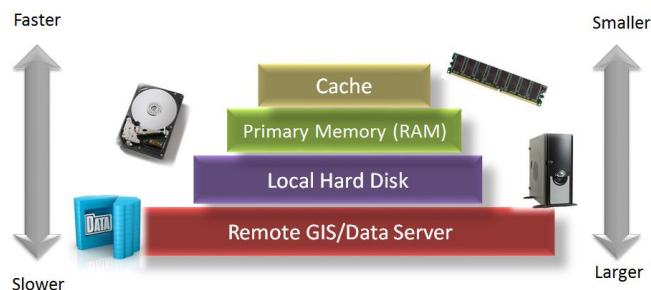


Figure 1-1 Data storage levels with corresponding speed and capacities

How data get through all these levels simply illustrated by an online map browsing application through a browser. The corresponding geographic data usually resides in remote web or Geographic Information Science (GIS) server and whenever data is requested by client, first remote server fetches the data from its local storage and then this data is sent to client computer which stores this data in its local hard disk. This data is being used by application, for given case, it is used to display 2D map on browser. To render map, this data is loaded into primary memory and then usually to Graphical Processing Unit (GPU) memory. As data travels through these layers, the speed of mediums increases, nevertheless the storage capacity is decreasing with higher layers.

Recent advances in computer hardware and software technologies increase the application requirements for using, analyzing and processing very large geo-spatial data [1, 2, 3]. Processing big geographic data is still crucial in a number of application domains like GIS, flight simulations and interactive 3D games [1]. As availability of geographic data increases and exceeds the size of main memory, I/O operations between primary (main memory) and secondary memory (hard disc storage) become an important bottleneck. To ameliorate this situation, caches have been used in a variety of layers mentioned above. In basic case, a system may employ a typical two-level memory architecture which consists of a small but fast cache and a relatively large but slower memory. Nowadays, caches are also being employed in GPUs. Independent from employed cache architecture, to have effective caches, the hit ratios should be increased. Thus, prefetching techniques are becoming critical to manage this data for all these kinds of systems through predicting which chunk of data will be needed next and fetch it into the cache.

Prefetching techniques have been used by different systems to handle this big data [4, 5, 6, 7, 8, 9]. In this study, the focus is on the prefetching itself. It is not only crucial for big data management in GIS but also for 3D visualization.

Internet mapping applications have been popular for the last decade. The internet, or in other words, World-Wide Web, permits anybody to access the vast quantities of information instantaneously. As performance increases with improvements in bandwidth and infrastructures, users continue to require less delays. In parallel, content providers continue to make greater demands on bandwidth. At this point, response time or user perceived delays become very essential for user satisfaction and productivity [10, 11, 12]. A widely-accepted study from Zona Research [13] provides evidence for the “8-second rule” in e-commerce (electronic commerce) which states that if a WEB site takes more than 8 seconds to load, the user is much more likely to become frustrated and leave the site. Thus, it is very important for geographic applications to provide a responsive interaction which is achieved by prefetching.

1.1 Problem

Prefetching is a process of loading user objects proactively before requested. More specifically, it mainly concerns about the retrieval of data from disk (out-of-core) into memory (in-core) to fulfill future application requirements. The primary purpose of prefetching is to ensure that at any one time, data required for application is already loaded into memory for further processing like analysis or visualization. Some existing techniques [14, 15] do nothing and let operating system does the actual fetching (paging) of data during runtime, but it is obvious that this approach is not feasible anymore.

Prefetching techniques usually request data in the form of geographic tiles to improve the efficiency. Therefore, most of the available approaches generate data tiles instead of requesting whole data [8, 16]. The tiling minimizes the initial response time and reduce the transfer time. The dealing with tiles is more manageable than dealing with whole data. Hence, most of the studies about prefetching use tiling. In addition to tiling, layering is also employed by prefetching approaches in such a way that when user goes into a coordinate where no detailed data is available, first of all, a lower detailed data is loaded and then more detailed data is provided [16]. To achieve an

optimum memory usage and high performance, determination of tile requests should be tackled carefully. A simple but efficient unique tile identification method is proposed for this purpose in this study which is described in Section 4.4.1. The tiling and layering concepts are illustrated in Figure 1-2.



Figure 1-2 The raster data tiling and layering

Prefetching is classified into two groups according to its employed location which are server and client sides [8, 17]. Although strategies used for these two group may differ from each other, their characteristics are similar.

Prefetching techniques are usually developed for specific type of applications and embodied in systems as a tool for caching. Most of GIS approaches and applications use prefetching in conjunction with 2D map operations like panning or zooming to predict the user's next behavior. Some of these techniques use offline historical data for this purpose [18], others such as [19] uses geographic vector data. There are also drastically different approaches that employ neural network-based methods to determine these candidate tiles such as [16], but scope is still 2D map navigation.

There are also prefetching studies which are conducted in computer graphics [20, 21]. These studies, on the other hand, usually employ visibility-based prefetching techniques and take observer field of view (FOV), view direction, movement direction and altitude into consideration. Underlying rendering infrastructure uses

these techniques to fetch the necessary elevation and map tiles for 3D terrain visualization.

The most prefetching approaches are usually do not take different type of applications and navigation behaviors from different domains into consideration. They are usually developed for specific type of applications such as 2D GIS or 3D visualization applications and crafted for specific navigation patterns. However, these approaches become insufficient for hybrid application types that possess capabilities from both of these domains. They also do not employ a feedback mechanism for changing application state in such a way that navigation behavior of application may also vary at run-time. As a result, prefetching technique should be updated so that poorly performing technique's contribution should be reduced and vice versa.

Current prefetching approaches solve prefetching problem for specific type of applications and domains. However, as the boundary between GIS, visualization and applications blurs, these approaches become insufficient for hybrid applications which possess capabilities from both of these domains. Digital moving map systems are one example of such applications. They provide 2D map, 3D terrain visualization and analysis capabilities. Most of these are running on avionics systems that have very limited resources. Although specific prefetching techniques perform well, for instance, for 2D map navigation, they do not perform well if user switches to 3D visualization mode and vice versa. For 2D navigation, the Retrospective Adaptive Prefetching (RAP) method [4] or other heuristic or probability based methods can be employed, but these are not sufficient or suitable for 3D navigation. Nevertheless, 3D simulations or digital moving map applications begin to make use of this geographic data and capabilities which are provided previously by GIS applications. Therefore, no single prefetching technique fits perfectly to all these types of applications and navigation behaviors. Nevertheless, most of these techniques may not perform well with varied patterns and capabilities due to not employing a feedback mechanism to response such changes.

To overcome these limitations, different prefetching techniques should be used together to provide a better prefetching performance which is the main motivation of this study. In this study, a priority based ensemble adaptive tile prefetching (EAP) approach is proposed which enables ensemble usage of prefetching techniques at the same time to fulfill different application caching requirements.

In the light of these facts, two important questions emerge about prefetching;

- Which raster tiles should be prefetched and in which order they should be prefetched?
- How can different type of prefetching techniques be combined to obtain a better prefetching performance?

Although some prefetching techniques are developed to illustrate the approach, the main focus of this study is on second question. The first question is thoroughly studied and many prefetching techniques have been developed for specific applications and domains. In this paper, an adaptive approach is proposed which contains an ensemble of prefetching techniques registered to be used according to corresponding application capabilities.

It is believed that a better prefetching performance can be obtained if a combination of prefetching techniques are used at the same time for applications that exhibit different capabilities. Even better hit ratios can be obtained if weight of these techniques are being set adaptively.

With proposed approach, prefetching performance increases are showed even for the applications that exhibit different characteristics. Moreover, as a result of using more than one prefetching technique, the order of tile requests are determined more precisely. Focus of this study is on client side even though it is possible to extend this approach to be used at server side.

1.2 Contributions

In this study, a priority based ensemble adaptive tile prefetching (EAP) approach is proposed which enables the group of prefetching techniques used together at the same time. A fuzzy logic based inference engine is developed to determine the contribution of each technique. This engine prioritizes the tile requests and tiles that reside at cache dynamically to obtain the optimum load and replacement order. With proposed approach, applications can also change criteria parameters manually at execution time through provided generic interface.

A generic prefetching model is developed which fits for most of the prefetching techniques. This model is described through well-defined steps, data forms and the term ‘criterion’ which is introduced to represent the prefetching and related operations such as requesting, cache replacement and evaluation. When current approaches are studied, although most of them come up with some steps of prefetching, concrete and well-defined steps of prefetching are not given or described.

The proposed EAP approach is implemented through a cross platform framework to let other users utilize EAP for different types of applications. Each application requires a different set of prefetching techniques to be used for this purpose according to application requirements. With this framework, different prefetching techniques can be experimented to find optimum set and also to observe the performance. The four different prefetching techniques are developed to be used as basis for 2D, 3D, analysis and Point of Interest (POI) applications. Besides, RAP [4] prefetching technique is implemented using given framework to illustrate possible other prefetching technique employment in EAP. These techniques are used for experimentation of a hybrid application.

In brief, this dissertation makes the following contributions;

- Ensemble usage of different prefetching techniques at the same time,

- Adaptiveness through employed fuzzy logic & online feedback mechanism,
- Better prefetching performance even with hybrid applications,
- More precise load order,
- Prefetching formalization,
- Easy utilization of EAP through generic cross-platform framework,
- Development of five example prefetching techniques which are 2D MapView, 3D FOV, Vertical Cross Section (VCS) Analysis, Retrospective Adaptive Prefetching (RAP) and Point of Interest (POI).

1.3 Thesis Overview

The introduction chapter gives a brief background information about problem and motivation of this study. Chapter 2 examines general prefetching concepts and related work about prefetching approaches. In this chapter, not only studies from GIS field, but prefetching techniques employed in other fields like 3D visualization are also given to illustrate the commonality of problem in various domains. Chapter 3 introduces the fuzzy logic which is used in adaptive part of EAP. Chapter 4 describes the proposed EAP process model and framework with all related concepts in detail. Chapter 5 presents the set of prefetching techniques developed for EAP in the scope of this study. Experimentations conducted about EAP to illustrate its performance and the results obtained from these experimentation are discussed in Chapter 6. Finally, the summary and future works are given in Chapter 7.

CHAPTER 2

PREFETCHING

In this chapter, background information about prefetching concept, its role and studies conducted about prefetching from different domains are described.

Prefetching is a process of loading user objects proactively before requested. The primary purpose is to ensure that anytime data required for user is already loaded into memory and available for further operations. It is also being employed to alleviate the data transfer costs that may occur as a result of I/O or network communication [22] in case of retrieving data from distant data servers.

Before delving into prefetching concepts, a simple usage of prefetching is illustrated below in Figure 2-1 where application required to load necessary map tiles into memory to render map.

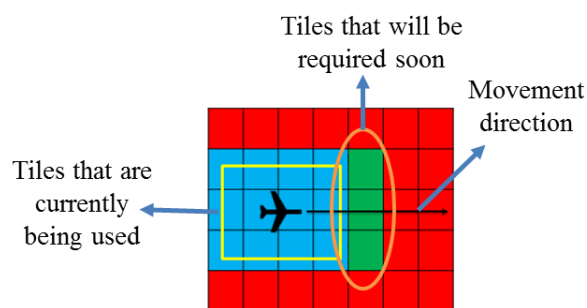


Figure 2-1 A simple usage of prefetching

The tiles represent the physical map tiles where blue parts represent the already loaded and rendered map tiles, red ones are not being loaded yet. The yellow lines show the visualized extent of map. In simplest scenario, as observer moving

horizontally along a path, it is predicted that the next tiles that would be required are the ones that are shown in green. What prefetching does is to load them into cache before map extent is being slided to that region and user always sees map flawlessly and experiences no load delays. Here, it is also worth to mention that prefetching mechanism is closely related with caching mechanism. The prefetched data usually does not being used by applications directly, and they are usually stored in a cache till application require that tile.

Prefetching also plays a very critical role in WEB browsing which becomes more popular in recent decades [23]. This study presented a prefetching approach to reduce the WEB latency experienced by web clients, by prefetching documents, before they are actually requested by web browser. The proposed idea is to keep the top ten popular documents for each web server, by this means, clients or proxy servers can prefetch only these popular documents without significantly increasing network traffic. The proposed study results show that this approach expects more than 40% of client requests and achieves close to a 60% hit ratio at the cost of increasing network traffic by no more than 10% in most cases [23]. Briefly, their approach is based on the cooperation of server and client-side entities as illustrated in Figure 2-2. More examples of such studies are given in section 2.1.

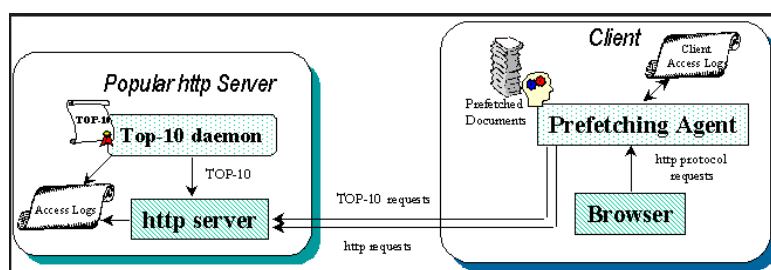


Figure 2-2 Cooperation of server and client-side entities for prefetching [23]

Prefetching is not a specific concept for geographic information systems, visualization or web technologies domain, it also being used prevalently by processors and file systems in such a way that Central Processing Unit (CPU)s and operating systems use prefetching to load instructions and data into CPU for a long

time [24]. This kind of prefetching is implemented either by hardware or software. Besides, database systems make use of prefetching in order to minimize the number of data fetches between the client and the server in a database management system [25].

The data that is being loaded by prefetching is completely based on the domain that prefetching is being used. There are various kinds of data from geographic elevation and imagery data to the WEB HTML pages. Special techniques are incorporated to improve performance of prefetching for some of these data. For instance, most of the geographic data is being tiled into smaller parts to reduce the user perceived delays and flawless user experience. An example of this tiling is used in well-known google maps [26]. It makes use of a tiling mechanism where first level (level 0) represents the whole world in a single 256x256 pixels tile (where the first 64 and last 64 lines of the tile are left blank) which is illustrated in Figure 2-3. The next level represents the world in 2x2 tiles of 256x256 pixels and this continues in powers of 2. The number of tiles in each level grows exponentially with the resolution level l. For instance, to cover the whole world with a pyramid of 20 resolution levels, approximately 3.7×10^{11} tiles are required which corresponds to petabytes of data.

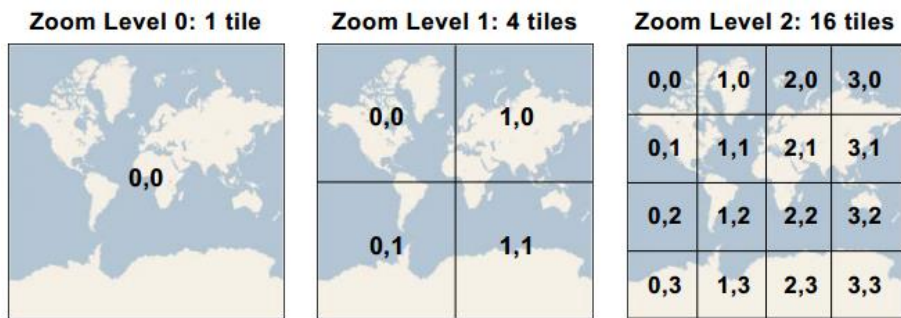


Figure 2-3 Google Maps tiling schema [26]

The underlying layout of data and how it is being indexed is also important for prefetching data request. For instance, a 3D visualization application may employ

octree like spatial data structures to identify efficiently the data that lies inside the prefetching view frustum.

Prefetching techniques can be classified according to their employment location which are server side and client side prefetching [8, 17]. The server side methods try to prefetch data according to previously initiated requests by clients and usually do not have knowledge about the context and application capabilities that initiated these requests. The client side prefetching, on the other hand, usually evaluates the current state of application, predicts the possibly required data in the near future and initiates prefetching requests to the server. It is also possible to employ a hybrid approach where both client and server side prefetching techniques are used at the same time.

In addition to client and server side usage, prefetching is also employed to load data that reside at local storage into main memory which is usually the case for client side GIS applications. Moreover, as the GPUs become more prevalent, the imagery or elevation data might be needed to be prefetched and loaded into GPU buffers from main memory. As the GPUs start to support this capability by hardware, some approaches started to make use of these in their application. Virtual texturing and mega texturing [27] are example of these approaches others of whom are given in detail at section 2.2.

Finally, prefetching techniques can also be categorized according to their nature. Although such categorization in detail is out of scope of this study, it is believed that mentioning some of such categories may help new prefetching researchers to follow previous studies. For instance, some prefetching mechanisms make use of historical data to determine candidate data for prefetching, on the other hand, some studies only make use current data for prediction. Probabilistic methods which also making use of techniques like Markov chains are also proposed in addition to lately introduced neural network based learning prefetching techniques [28]. There are also heuristic methods which are based on the idea that there is no standard way of determining the user's behavior or finding out exact solutions for prefetching may never exist [4].

Some heuristics may improve the performance of prefetching through using other parameters such as historical data.

No matter which kind of prefetching is proposed, most of them makes use of application specific parameters or attributes for their prefetching. Nevertheless, each of these approaches only focuses on one specific technique and no ensemble mechanism or approach is proposed yet which is also done with this study.

After having discussed the core prefetching concepts, now studies conducted about prefetching are given in this section. As noted before, prefetching is being used by various techniques like out-of-core rendering or Computer Aided Design (CAD) visualization besides the wide-spread GIS usage. As a result of this, related prefetching studies from both GIS and visualization domains are given under two categories to illustrate the current approaches. Finally, for the sake of completeness, prefetching studies from different domains are also briefly described in a separate category.

The most important motivation of scanning various literatures is to show that although requirements or capabilities of applications from these domains are different, the aim of these studies is same which is to make sure that data is loaded before requested. These studies are also important to see how prefetching can be used for different kinds of applications. Finally, it also important to note that the boundaries among these two different fields are blurred and now simultaneous usage of these capabilities is possible.

2.1 Studies from GIS Domain

First of all, some techniques from GIS field are going to be described. Most of these studies focus on web prefetching. The main reason for this is the latency that occurs over internet connection as in IO, popularity and prevalence of web technologies. Although individual computer processing power increases drastically, the network

speed is not increased in same pace. In web applications, quick responses usually depend on less process on server side and nowadays less process is mostly achieved by the caching and prefetching policies. So these studies mainly focus on minimizing processing at server side, reduce the requests initiated by clients, and improve the hit ratio to provide low retrieval latency for user as described in [17]. Mobile studies are also summarized and mentioned under this section.

The first study [29] performed a survey about characteristics of web objects that are used for replacement. The cache replacement, as is described in section 4.4.4, is very similar to prefetching and same approaches can be used for both prefetching and replacement in reverse logic. For instance, the web object which is selected according to Least Recently Used (LRU) strategy can be converted to Most Recently Used (MRU) way. These techniques are given here to show how these techniques and parameters can be used for prefetching.

The study classified the following characteristics of web objects for replacement strategies which can also provide insights for other prefetching methodologies.

Table 2.1 Characteristics of web objects [29]

Identifier	Description
s_i	Size of object i
t_i	Last request time of object i
T_i	Time since last request to object i
f_i	Number of past requests to object i
l_i	Access latency for object i
c_i	Cost to fetch object i from its origin server; c_i has a more general meaning than l_i (e.g. number of network hops, latency).

- Recency: the time since the last reference to the object,
- Frequency: the number of requests to the object,
- Size: the size of web object,

- Cost: the cost to fetch the object from the server that resides.

According to employed characteristics, the strategies can be classified as recency-based, frequency-based, recency/frequency-based, function-based and randomized strategies [29].

Recency-based strategies focus on temporal locality of reference observed in WEB requests. These are similar to the well-known LRU strategies, which are removing the least recently referenced object. The other mentioned study, Pyramidal Selection Scheme (PSS) [30], uses a pyramidal classification of objects depending upon their size, and objects in each group are maintained as a separate list. Only the least recently used objects in each group are compared with each other, and the object which maximizes the product of its size and the number of accesses since the last time was being requested.

Frequency-based strategies use the fact that popularity of WEB objects is related to their frequency values into consideration. These strategies make use of the LFU (Least Frequently Used) strategy, which removes the least frequently requested object. Recency/frequency-based strategies combine recency and frequency information for prefetching.

The next study is from Dong-Joo Park and Hyoung-Joo Kim who presented a prefetching policy for retrieving large objects in WEB GIS applications [5]. They made an assumption on how the user accesses the geographic objects on the map and stated that it is very likely that the user chooses the neighbor objects around a certain central object which is called “callback object”. In other words, the access pattern of the user is determined according to the “spatial locality”. Under this assumption, they proposed a Hilbert Curve based prefetch algorithm where they employed Hilbert Curve for selecting a set of candidates based on the current callback object. Such candidates will have more probability to hit the client cache when the next callback object is requested. In this approach, map is divided by the Hilbert Curve and each

cell is assigned with the values of that Hilbert Curve as shown in the Figure 2-4 below.

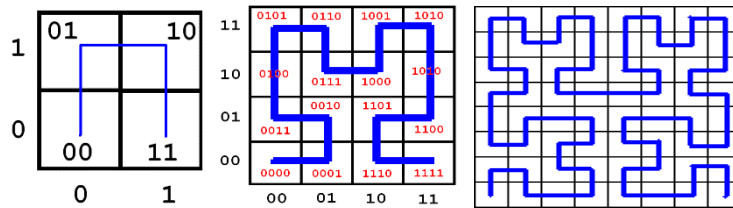


Figure 2-4 The first three steps of Hilbert Curve [5]

An array of Hilbert Curve values are defined and each object is placed in this array with their corresponding Hilbert Curve values. The candidates which can be selected either dynamically or statically are selected from the left and right side of the callback object in the array. To give an example usage, assume that the location of the callback object is “ i ” and callback object is represented as “ O ”. In static way, a value called “window size” is chosen and if we say window size is “5”, the objects $O_{i-2} \dots O_{i+2}$ are selected as candidate objects. In dynamic approach, the window size is adjusted by the “spatial locality by distance” (SLD). The candidate objects are still selected from the left and right side of the callback object, but this time, instead of taking static number of objects, all the objects whose distance from the callback object is smaller than the SLD are chosen as the candidate objects. As a result of having varying number of candidates as the SLD value changes, the window size is dynamic. Also, a formula to determine an appropriate SLD value is defined. Through the experimental results, they show that the performance of the dynamic scheme is close to the static scheme; on the other hand, the network traffic of the former is lower than the latter.

The third study is from Dong Ho Lee et al. who proposed two prefetching techniques [28]. One of these techniques is probability-based and it makes use of the location of the tile to predict the next navigation. It is stated, if the tile is located near to the upper border of the view extent, then the user is probably to navigate to an upper tile than the lower one. Moreover, it also takes the zooming levels of the current view extent

into consideration too. The probabilities of all navigations are calculated and top “t” (number of tiles to prefetch) tiles with the highest transition probabilities are selected. The second technique described makes use of heuristic rather than the probability as previous method. It is called previous-k-movement approach. In this technique, rather than the current position as employed in the previous technique, former actions of the user determine the next movement. A Neighbor Selection Markov chain (NSMC) is created to obtain the neighbor selection probabilities that can be applied to a general tile. A state on NSMC presents a sequence of direction vectors that gives the tile selection history where this direction vector denotes a move from one tile to another and an example of that is shown below in Figure 2-5.

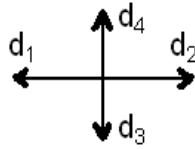


Figure 2-5 Direction vector on a 2-dimensional space [28]

In this method, the probabilities of previous navigations are put on a Hilbert Curve and next tiles are predicted according to patterns on this curve. The main difference between this method and Hilbert Curve based prefetch algorithm [5] is; this method makes predictions by using the past n moves before a specific tile instead of just using that specific tile.









Another study from GIS domain is Retrospective Adaptive Prefetching (RAP) approach [4]. The RAP mainly focuses on map serving problem in GIS domain. Since most of the GIS services provide their geospatial data as basic image formats like PNG and JPEG, constructing those images and transferring them over the internet are costly operations. Caching the responses of the requests on the client side is the most commonly implemented solution. However, this method is not adequate by itself. Besides caching the responses, predicting the next possible requests of the client and updating the cache with the responses for those requests provide a remarkable performance improvement.

The prefetch algorithm provided in RAP make use of a heuristic approach for retrieving the next possible tiles of a given tile according to previous navigations. Number of previous navigations, history depth, is determined as a configurable parameter. As depth increases, the accuracy of the result increases as well, however, the number of tiles to prefetch is not affected by the depth number. The only factor that affects the number of tiles to be prefetched is the pattern of the former navigations in history. Namely, if the user stays in a stable path during navigation, the algorithm assumes that she will probably keep on moving on the same path in the next moves.

Otherwise, for instance, if the user makes random navigation on 2D map, the estimation of the algorithm will not be effective and hit ratio of the prefetching cache reduces. The algorithm given has a grading formula for determining the number of neighbor tiles with their relative locations according to source tile. Then, all these values are combined and divided into total differences. Namely, for instance, each Easting has a “difference value” which is normalized according to the hit ratio obtained by the prefetched tiles and those tiles are calculated with the help of that Easting value. The order of tiles that will be prefetched and loaded according to these values on the other hand is static which is shown in Table 2.2. The study compares their results of the proposed algorithm with previously mentioned methods which are Hilbert Curve Based Prefetching (HCBP) Algorithm [5, 28] and previous-k-movements (PKM) method.

An example navigation used for RAP is given in Figure 2-6. The $\langle 2, 2 \rangle$ cell is the center of the initial map extent which consists of 9 tiles. The square around the last tile represents the map extent of the final state. 2 bright arrows shows the first two navigations which are removed from the history because they are expired when user comes to $\langle 8, 5 \rangle$. By expired, it is meant that 5 (history depth) navigations are passed after these two tiles and because of that they are no longer needed for calculations.

Table 2.2 Neighbor tiles to prefetch according to Easting and Southing

TILES TO PREFETCH	EASTING	SOUTHING
	1	0
	-1	0
	0	1
	0	-1
	1	1
	1	-1
	-1	-1
	-1	1

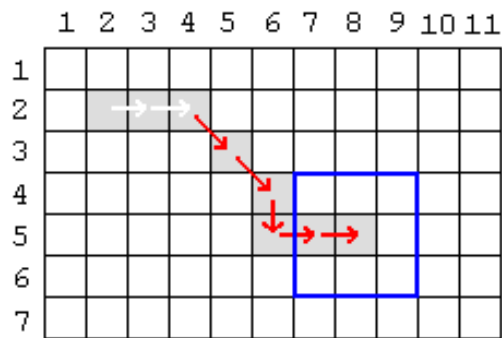


Figure 2-6 Sample navigation scenario when history depth is 5

Each step of grading formula according to above navigations used for RAP execution is shown in Table 2.3.

Table 2.3 Execution of grading formula for sample scenario with history depth 5

Step	Previous Tile	Current Tile	Hit Ratio	Easting	Southing
1	2 , 4	3 , 5	0.692	$(5-4)*0.652 = 0.652$	$(3-2) * 0.652 = 0.652$
2	3 , 5	4 , 6	0.652	$(6-5) * 0.652 = 0.652$	$(4-3) * 0.652 = 0.652$
3	4 , 6	5 , 6	1	$(6-6) * 1 = 0$	$(5-4) * 1 = 1$
4	5 , 6	5 , 7	1	$(7-6) * 1 = 1$	$(5-5) * 4 = 0$
5	5 , 7	5 , 8	1	$(8-7) * 1 = 1$	$(5-5) * 1 = 0$
Total				3.304	2.304
Average:				$3.304 / 4 = 0,826 \rightarrow 1$	$2.304 / 3 = 0,768 \rightarrow 1$

A relatively different approach is proposed in [31]. They say that significant improvement is achieved through employing background geographic information in addition to previous access patterns that are used by previous studies for prediction. A regressive model is proposed to predict probable areas that user will probably request soon. This is performed according to spatial cross-correlation between an unconstrained catalog of geographic features and a record of past cache request. The approach is experimented through Spain nation-wide public WEB map services. The proposed system architecture of this approach is depicted in Figure 2-7.

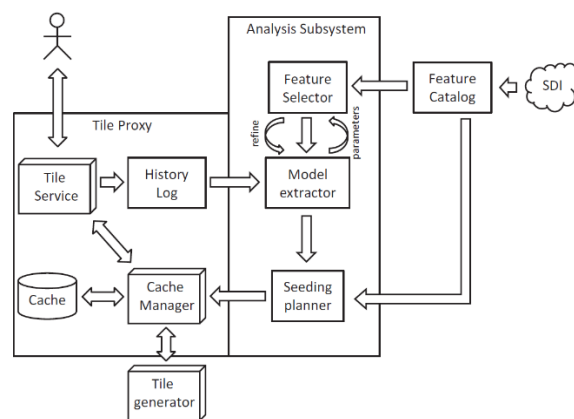


Figure 2-7 System architecture of [31]

Similar to previous approach, Quinn and Gahegan proposed a model for determining popular areas for caching by considering geographic features [18]. Their model take variables found in previous research into consideration for WEB map users, such as populated places, major roads, coastlines, and tourist attractions. They said that the content that users probably will request can help server administrators know which tiles to create in advance and to include in their server-side caches of map tiles. So their primary aim is to predict these areas of the map that will be most often viewed and should therefore be cached.

In other study [8], a probability-based tile pre-fetching with a collaborative cache replacement algorithm for WEB geographical information systems (WEB GISs) are proposed. The basis Web GIS architecture used for this approach is shown in Figure 2-8. It can be seen, in this approach prefetching is employed at server-side. The study also summarize how this architecture and prefetching for such WEB GIS systems are working.

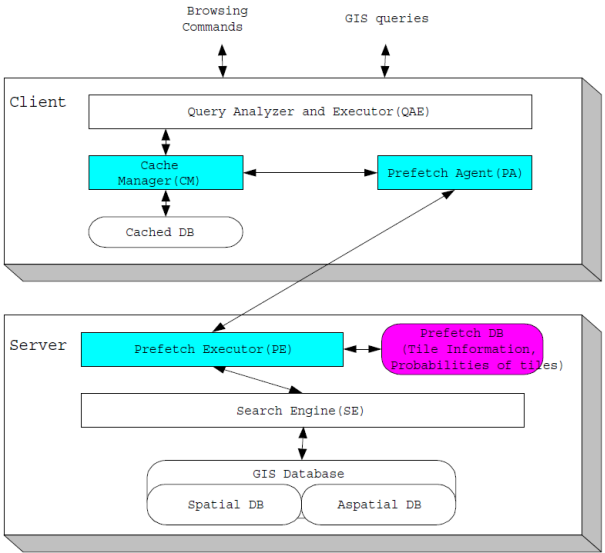


Figure 2-8 Abstracted architecture of Web GIS systems [8]

The server collects and maintains the transition probabilities between adjacent tiles. With these probabilities the server can predict which tiles have the highest probability

of access in next time than others, based on the global tile access pattern of all users and the semantics of query. The proposed cache replacement algorithm determines which tiles should be replaced based on the estimated future access probabilities. The proposed tile pre-fetching algorithm is employed with the cache replacement algorithm to improve the response time for user requests in WEB GIS systems.

In [32], the web application Hotmap is proposed. This approach analyzes log requests from the Microsoft's Live Search Maps service and visualizes the number of requests of each tile accurately on a map. These studies reflect that several features which drive most people's attention, like shorelines, roads and populated places, can be identified.

There are also studies that make use of neural networks for prefetching [16]. In this study, a neural network based method which takes background geographic information into consideration is proposed. It is stated that map tiles are not visited randomly and behavior of users is closely related with semantics of features combined in map tiles. History of service is also taken into consideration. Neural network is used to assign non-binary priorities automatically to map tiles according to performed training. The training is performed through a collection of past accesses with a catalog of geographic features, such as runways, highways or harbors, as predictor variables.

It is obvious that all these approaches are focusing on some parameters and it would be very effective to combine or use all these collaboratively which become the most important motivation for this thesis.

2.2 Studies from Visualization Domain

Now, prefetching mechanisms employed in visualization techniques are described here to show how prefetching is being used this domain with corresponding techniques. Although these studies are not directly aiming prefetching problem, they

employ prefetching as part of overall problem. Most of these studies are about out-of-core rendering. Prefetching is active where the underlying model or data is being loaded into memory part by part. So first a background information about out-of-core rendering is given prior to describing the related work.

Out-of-core rendering

The purpose of out-of-core rendering studies is to render a model without loading the entire model into memory [14, 33]. As the terrain data become more and more and exceeds the size of main memory, I/O operations between primary (main memory) and secondary memory (hard disc storage) become an important bottleneck. A disk access is about one million times slower than an access to main memory. A simple management of external memory, e.g., with standard caching, may affect and in fact decrease the performance. Some computations over these datasets might not be local and may require large amounts of I/O operations. At this point, out-of-core algorithms and related data structures manage how data are loaded and how they are stored. Typically, data is organized into some hierarchy and then chosen for rendering based on a user-specified search criterion. Therefore, only small amounts of the data are being processed at any one time, alleviating the need to process all the data at once. However, much of this work is focused on rendering a specific region of this huge data and not the whole data itself, using only one dataset at a time.

On top of this problem, the amount of graphics processing unit (GPU) memory available to desktop machines is not capable of storing all these datasets at runtime. Therefore, only a few of the datasets exist in memory (in-core) at any one time, specifically the datasets which are being used in rendering the visible terrain, while the rest are stored on the hard drive (out-of-core) until needed. The problem is how to determine if the data exists on the hard drive and not in the main system memory of the computer. Some approaches are making use of only a spatial subdivision hierarchy to alleviate this problem as datasets can be grouped based on their relative geographic locations. Other approaches make use of multi-resolution models and intelligent prefetching mechanism for this purpose. This model can then remain in

memory without any data loaded simply to determine which datasets should be tested for rendering when the user makes a search query. As the number of datasets is large, creation of the hierarchy should happen in a preprocessing step and written to the hard drive, obviating the need for the runtime version of the algorithm to recreate it.

The problem of out-of-core visualization is also studied from several aspects, which can be broadly classified into three closely interlinked sub-problems that should be addressed. These are listed as follows; terrain rendering, data query and pre-fetch, which correspond, respectively, to processing data to be displayed for terrain visualization, identification and organization of data to be retrieved, and retrieving data into in-core memory in anticipation of needs for processing in the near future.

The terrain rendering sub-problem arises from the complexity of out-of-core data which makes it infeasible to render the geometry of the entire scene as mentioned above. State-of-the-art approaches focus on reducing scene complexity by reducing the amount of geometry to be processed by the graphics pipeline, while maintaining good visual fidelity which will be tackled through employment of multi-resolution model and Level of Detail (LOD) in this approach.

The query sub-problem involves the layout of data and the indexing of data so that region queries can efficiently identify this data in the view frustum. Contemporary approaches use spatial data structures such as quadtrees, octrees and R-trees that perform spatial partitioning and data clustering in the scene. These structures do not only provide fast scene queries through their hierarchical properties to localize data currently in view, but also help in other respects like collision detection and analyses.

The prefetching sub-problem which is also important for this thesis mainly concerns the retrieval of data from disk (out-of-core) into memory (in-core) to fulfill future processing requirements. The problem is to ensure that at any one time, data required for terrain visualization is already loaded in memory. After having described out-of-

core rendering now other studies that make use of prefetching are given with brief descriptions.

Prefetching Studies in Visualization Domain

The first study that makes use of out-of-core rendering is aiming to visualize very complex 3D models at interactive rates [7]. In this study, a subset of this complex model is selected as preferred viewpoints and partitions the space into virtual cells. Each cell contains near geometry, rendered using levels of detail and visibility culling, and far geometry, rendered as a textured depth mesh. The system then automatically balances the screen-space errors resulting from geometric simplification with those from textured depth-mesh distortion. Each new viewpoint sent into the pipeline is passed both to the prefetcher and to the rendering pipeline. The viewpoint determines what geometry and textured depth mesh (TDM)s are retrieved from disk. A speculative prefetching is implemented as an asynchronous activity in this study. The virtual cells are organized into a graph structure to facilitate prefetching. They maintain a priority queue of geometry that would be needed soon and priorities are assigned according to their cell distances. Then this prefetcher activity uses this queue for loading data from disk. An overview of their system is given in Figure 2-9.

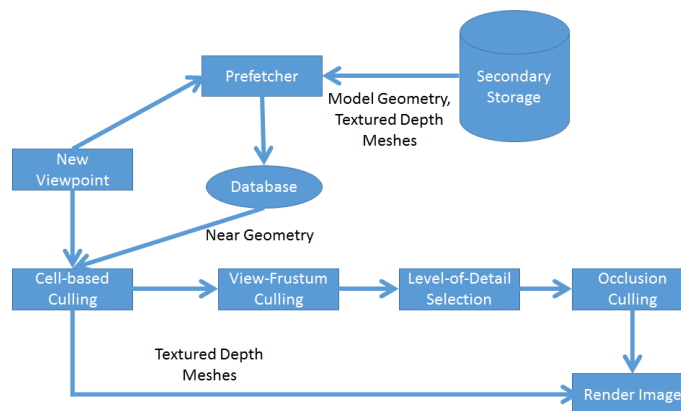


Figure 2-9 System pipeline of [7]

Another study is from Varadhan and Manocha [20]. There were also studies that make use of out-of-core rendering with hierarchical levels of detail. In this study, an external memory algorithm for fast display of very large and complex geometric environments is given. The model is represented as a scene graph and additionally, different culling techniques for rendering acceleration are employed. The algorithm also uses a parallel approach to render the scene as well as fetch objects from the disk in a synchronous manner through simple threading. The memory overhead of algorithm is output sensitive and is typically tens of megabytes. In practice, this approach scales well with the model sizes, and its rendering performance is comparable to that of an in-core algorithm. The algorithm was also employed to large gigabyte sized environments that are composed of thousands of objects and tens of millions of polygons.

Their system implements a priority-based pre-fetching algorithm which prioritizes objects in the scene based on some screen space error metric as well as their relative positions from the viewer's line of sight. The pre-fetching mechanism employed here fetches multiple LODs for each object according to their priority in order to ensure that there is a high hit rate even when the object switches from one LOD to the next. The approach employs an expanded view frustum to reduce the page fault rate and the objects in the expanded view frustum are also prioritized according to their distance away from the actual view frustum. More specifically, they take field of view (FOV) parameter as basis for their approach and calculate prefetching data as the union of possible FOVs for given time interval and determined set of parameters like viewer's heading and motion parameters. Then, it takes this into consideration for LOD selection and geo-morphing. The employed scene graph representation is given in Figure 2-10.

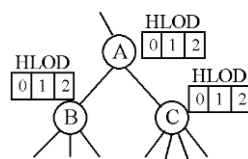


Figure 2-10 Scene Graph Representation [20]

Another study paper that will be mentioned here is [21]. The system uses out-of-core preprocessing algorithm and a multi-threaded out-of-core rendering approach. The out-of-core preprocessing algorithm is incremental and fast, and it builds an on-disk hierarchical representation for a model larger than main memory. The out-of-core rendering approach uses multiple threads to overlap rendering, visibility computation, and disk operations. A rendering thread uses a from-point visibility algorithm to find the nodes of the model hierarchy that the user sees, and sends fetch requests to a geometry cache, which reads nodes from disk into memory. The algorithm presented here builds an out-of-core on octree whose leaves contain the geometry of the model. To store the octree on disk, study saves the geometric contents of each octree node in a separate file, and creates a hierarchy structure (HS) file. The HS file has information about the spatial relationship of the nodes in the hierarchy, and for each node it contains the node's bounding box and auxiliary data needed for visibility culling. Figure 2-11 show the out-of-core rendering approach of the iWalk system. For each new camera (a), the system finds the set of visible nodes using either approximate visibility (b), or conservative visibility (c). For each visible node, the rendering thread (d) sends a fetch request to the geometry cache (i), and then sends the node to the graphics card (e). The look-ahead thread (g) predicts future cameras, estimates the nodes that the user would see then (h), and sends prefetch requests to the geometry cache (i).

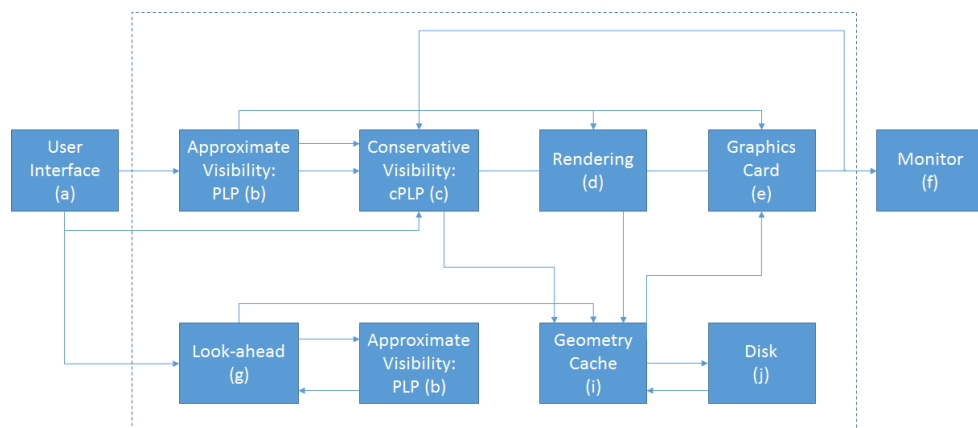


Figure 2-11 The out-of-core rendering approach of the iWalk system [21]

For visibility determination, it uses prioritized layered projection (PLP) algorithm. At runtime, the iWalk system uses this PLP algorithm to estimate the nodes potentially visible from the current view frustum (outlined in yellow). The transparent color of each node indicates the projection priority of the node. An example usage of this algorithm is illustrated in Figure 2-12.

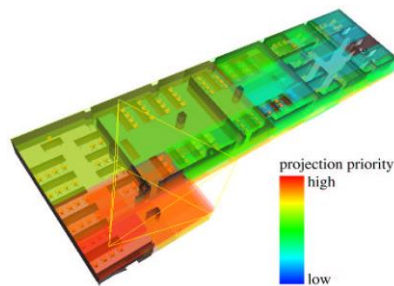


Figure 2-12 A section of model that colored according to PLP [21]

The other one is from Lindstrom and Pascucci [14] who proposed a new LOD scheme along with a data layout and indexing scheme using interleaved quadrees. Although, there is no explicit prefetching, the operating system handles the necessary paging of data. In the area of scientific visualization, Cox and Ellsworth [33] mentioned the use of application specific knowledge to control prefetching, loading and unloading of data for better performance.

The last study [34] state that visual exploration tools typically do not scale well with huge data sets, partially because being interactive necessitates real-time responses. It is observed that interactive visual explorations exhibit several properties that can be exploited for data access optimization, including locality of exploration, contiguous queries, and significant delays between user operations. Having considered these characteristics, they applied semantic caching of active query sets on the client side to exploit some of the above characteristics. They also introduced prefetching strategies, each of which exploits characteristics of their visual exploration environment. The result of this study then incorporated into XmdvTool which is a

public-domain tool for visual exploration of multivariate data sets. Experimental studies with synthetic as well as real user inputs are conducted and their results demonstrate that these proposed optimization techniques achieve significant performance improvements.

It can be noted that different from the GIS related studies, the most of the visualization techniques focus on visibility based prefetching. Therefore, visibility should be co-operatively used with other prefetching techniques to also satisfy 3D visualization application requirements beside GIS capabilities which is satisfied through EAP approach.

CHAPTER 3

FUZZY LOGIC

This chapter provides background information about fuzzy logic which is used in adaptive part of EAP. In this section, necessary information about fuzzy logic and related concepts are examined. How fuzzy logic being used in EAP is described in Chapter 4.

Fuzzy logic is a form logic which employs the partial values of truth instead of exact values as in traditional logic [35]. It is described as an extension to traditional logic for dealing with the knowledge representation problem in an environment of uncertainty. It is a form of logic where underlying modes of reasoning are approximate rather than exact. In traditional logic, values are either true or false. However, fuzzy logic permits degrees of truth. For instance, in traditional logic hot would be represented as discrete 1 value and cold would be described as 0 value as shown in Figure 3-1. In this figure, no value can be used to represent warm or cool temperatures.

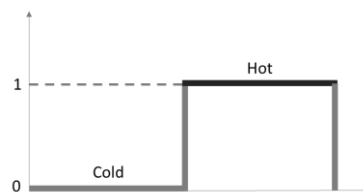


Figure 3-1 Traditional logic representation of a discrete temperature value

Different from traditional logic, the fuzzy logic describes in-between data values. The values represent the degree of truth through given ranges like 0 to 1 where 1 represent absolute truth or maximum and 0 represents the absolute false or minimum values.

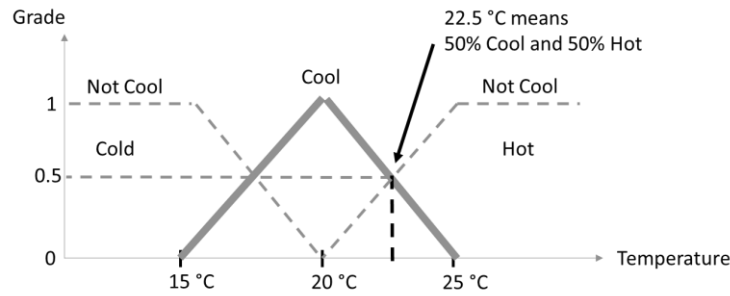


Figure 3-2 Fuzzy logic representation of temperature values

The Figure 3.2 represents the fuzzy logic version of same temperature values where cool temperature is also shown. Here 1 grade value represents the perfect cool air temperature. Any temperature below 15 °C considered as cold and temperature above 25 °C considered as hot. These ranges are represent the not cool temperatures. In addition to these representations, fuzzy logic consider 22.5 °C as 50% cool and 50% hot which shows a level of coolness.

In fact, fuzzy logic stems from human reasoning, which are usually based on approximate reasoning [35]. Moreover, it is described as a rule-based decision-making technique, employed for expert systems and process control, which primarily aims to emulate the heuristic, rule-of-thumb approach of human reasoning to many problems. For instance, fuzzy logic based temperature representation is used by a person to decide on wearing which kind of clothes before going out. According to current temperature, the type of clothing is being decided. As illustrated in Figure 3-3, at 20 °C short-sleeved shirt and pant can be worn. However, as temperature falls to 17.5 °C, a long-sleeved shirt instead of short one should be worn. If temperature decreases more like 15 °C, then a jacket might be worn additionally. It should be noted there could be more than one inputs to fuzzy logic as will be illustrated in following sections.

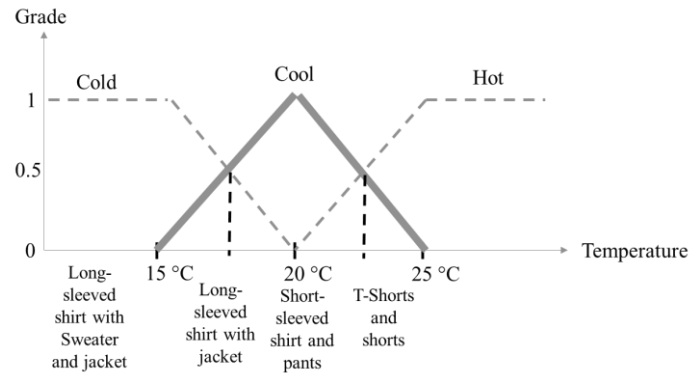


Figure 3-3 Fuzzy logic graph that illustrates the clothing choice according to temperature

Now on some historical information about development of fuzzy logic is going to be given. At the beginning of 20th century, first studies about fuzzy logic is conducted by Bertrand Russell as analyzing a Greek paradox which is given below [36];

“A Cretan asserts that all Cretans lie. So, is he lying? If he lies, then he is telling the truth and does not lie. If he does not lie, then he tells the truth and, therefore, he lies.”

When the given phrase is analyzed, in either case all Cretans lie or all Cretans do not lie which lead to a contradiction. Because both statements are both true and false. This become a basis for Russell’s study in such a way that he found that this paradox can also be applied to the set theory used in discrete logic where statements must either be totally true or totally false and this lead to areas of contradiction. In fact, fuzzy logic is based on this problem in traditional logic through allowing statements to be interpreted as both true and false. Therefore, when we apply fuzzy logic to given Greek phrase, we obtain that Cretans tell the truth 50% of time and lie 50% of time.

In addition to Russell study, another logician named Jan Lukasiewicz also started studying on multivalued logic which in fact created fractional values between logic 1 and 0 values. Then a quantum philosopher, Max Black, applied this multivalued logic to lists, drew the first set of fuzzy curves, and called them vague sets in his article [37]. About twenty years later, Lotfi Zadeh published a paper which was entitled as

“Fuzzy Sets” which in fact gave the name to the field of fuzzy logic [35]. In that paper, Zadeh applied Lukasiewicz’s logic to all objects in a set and deduce a complete algebra for fuzzy sets. As a result of his contributions, he is also considered as father of modern fuzzy logic.

After having given background information about fuzzy logic, now fuzzy logic system (FLS) is going to be described. FLS is defined as nonlinear mapping of an input data set to a scalar output data [38]. FLSs usually consists of three steps which are fuzzification, fuzzy processing (fuzzy rules, inference engine) and defuzzification. Overall process of a FLS can be described as illustrated in Figure 3-4. First of all, a crisp set of input data are gathered and converted into a fuzzy set using fuzzy linguistic variables, labels and membership functions. This step is known as fuzzification. Afterwards, an inference is made based on prepared set of rules which is known as fuzzy processing. Finally, obtained fuzzy output (FO) is mapped to a crisp output using the membership functions, in the defuzzification step.

FLS Process Algorithm:	
1.	Define the linguistic variables (at the beginning)
2.	Construct the membership functions (at the beginning)
3.	Construct the rule base (at the beginning)
4.	Convert crisp input data to fuzzy values using the membership functions (fuzzification)
5.	Evaluate the rules in the rule base (fuzzy processing)
6.	Combine the results of each rule (fuzzy processing)
7.	Convert the output data to non-fuzzy values (defuzzification)

Figure 3-4 Fuzzy logic system process algorithm

Overview of a fuzzy logic system is depicted as in Figure 3-5.

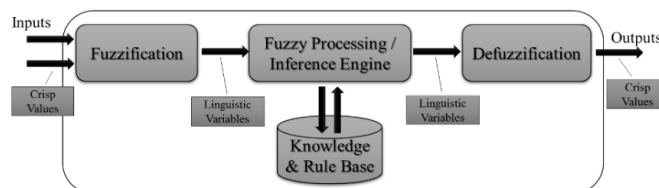


Figure 3-5 An overview of a fuzzy logic system

In following sub-sections, these steps and terminology related with fuzzy logic is going to be examined.

3.1 Fuzzification Step

Fuzzification step is responsible for converting input data into fuzzy membership; in other words, linguistic variables or labels [35]. A linguistic variable is one that uses linguistic values such as low, medium, and high. This is required to process rules that are defined in terms of linguistic variables. These variables are the input or output variables of the system whose values are words or sentences from a natural language, instead of numerical values. A linguistic variable is generally decomposed into a set of linguistic terms. For instance, hot and cold terms are being used to qualify the real life temperature values. These are used as linguistic variables of the temperature. Then these variables are further decomposed as too-cold, cold, warm, hot, and too hot. Each of these covers a portion of overall values of temperature.

These input values are analyzed according to user-defined charts, which are called membership functions and used to group received input data into fuzzy sets. It is used to quantify a linguistic term. An example of membership function is given in Figure 3-6. These functions are usually real-valued functions 0.0–1.0 and range outputs according to how well input fits into each function. The membership values should be 1 at the center of the set, in other words, for those members that definitely belong to the given set. These functions are defined by corresponding linguistic variables, which are called labels [36].

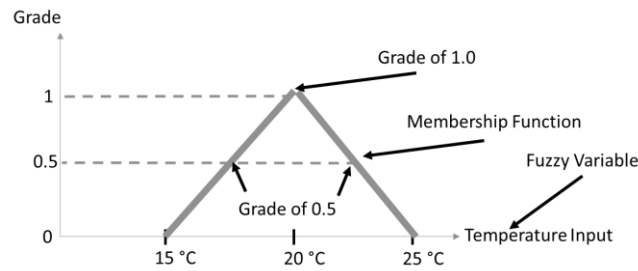


Figure 3-6 Membership function chart

The membership functions may have various shapes, depending on the data set such as S, Z, Δ , and Π [39]. The shape of the membership function is context dependent and it is usually chosen arbitrarily according to the user experience [38]. Some examples of different shapes of membership functions are depicted in Figure 3-7.

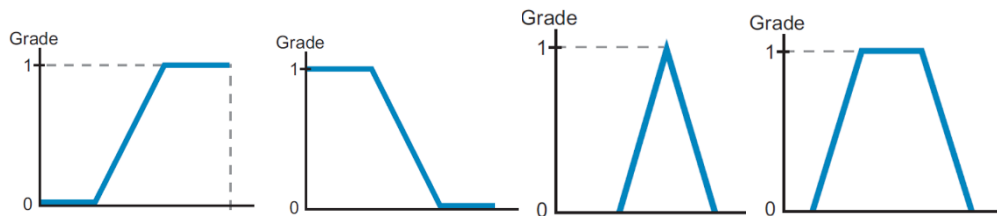


Figure 3-7 Different membership function shapes. S, Z, Δ , and Π shapes are illustrated from left to right [39]

It should be noted that a significant characteristic of fuzzy logic is that a numerical input does not have to be fuzzified using only one membership function. A value can belong to multiple sets at the same time. For temperature fuzzy set, a temperature value is considered as both cold and too cold at the same time with corresponding degree of membership. All these functions form a fuzzy set. Figure 3-8 shows a fuzzy set with five membership functions. Although most of the fuzzy sets have an odd number of labels, a set can also have an even number of labels depending on how the inputs are defined in relationship to the membership functions.

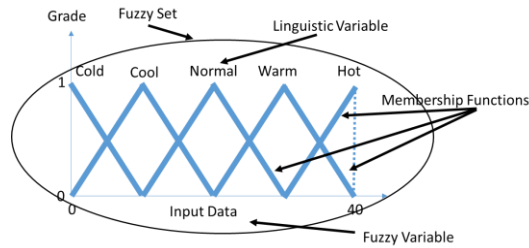


Figure 3-8 Example temperature fuzzy set with five \wedge shaped membership functions

3.2 Fuzzy Processing Step

The fuzzy processing component is primarily responsible for analyzing the input data that is defined through membership functions and arrives at a control output. This is performed by mapping from input fuzzy sets into output, according to the information stored in the rule base. After the inputs are processed as previously mentioned, the corresponding input fuzzy sets are passed to this component, which processes the current inputs using the rules retrieved from the rule base.

With fuzzy logic, most complicated problems can be formed through a collection of simple problems and can, therefore, be easily solved. It uses a reasoning with a set of rules which can also be called inferencing process. Each of these rules is in IF-THEN form such as “IF X_1 is A_1 and ... and X_n is A_n THEN Y is B .” An example set of rules that are created for clothing decision given in Figure 3-3 are given in Figure 3-9. The fuzzy rules with two inputs are also represented in a matrix form. For example, a system that have two inputs X_1 , X_2 with three terms and one output Y_1 are represented by a 3 x 3 matrix with 9 rules. This representation is preferred rather than the list one because it makes it easy to represent all the rules for a system.

- IF temperature is 20°F (grade 1–100% cool),
THEN wear short-sleeved shirt and pants.
- IF temperature is 17.5°F (0.5 cold, 0.5 cool),
THEN wear long-sleeved shirt and pants.
- IF temperature is 15°F (0.25 cool, 0.75 cold),
THEN wear long-sleeved shirt with a sweater and long pants.

Figure 3-9 Example rule set for clothing decision

A rule can have several input conditions that are logically linked in either with an AND or an OR relationship to trigger the rule’s outcome as shown in Figure 3-10. According to activated rules, the control output is generated based on the THEN part of these rules. A rule is activated if its input conditions, i.e., IF parts, are satisfied. There might also be cases where more than one rule is activated when all conditions are evaluated. After one or more rules are activated, outputs of each rule are combined into a single fuzzy set, which are considered as an outcome. This is the collection of one or more output membership functions, again with labels. In a fuzzy system, usually, there are many rules which corresponds to multiple IF conditions as shown in Figure 3-11.

		Condition		Action
Rule 1:	IF	$A_1 \text{ AND } B_1 \text{ AND } C_1$	THEN	Y_1
Rule 2:	IF	$A_2 \text{ OR } B_2$	THEN	Y_2
Rule 3:	IF	$(A_3 \text{ AND } B_3) \text{ OR } C_3$	THEN	Y_3

Figure 3-10 Rules with multiple input conditions that are logically linked with AND and OR relationships

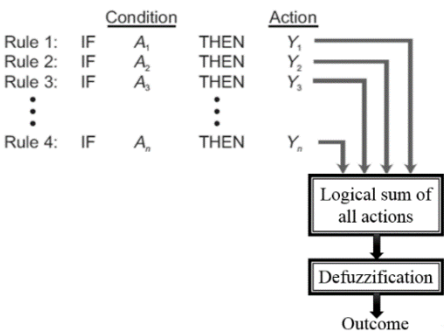


Figure 3-11 A fuzzy set with multiple rules

For evaluation of fuzzy rules that are mentioned above and the combination of the results of these rules, the fuzzy set operators are being used. It should be noted that the fuzzy set operators are different than traditional logic set operators. Let ϕA and ϕB be a membership functions which are defined for fuzzy sets A and B over universal set U. The three common operators (union/or, intersection/and, complement/not) are given in below.

$$\varphi A \cup B(x) = \max(\varphi A(x), \varphi B(x))$$

$$\varphi A \cap B(x) = \min(\varphi A(x), \varphi B(x))$$

$$\varphi \text{not } A(x) = 1 - \varphi A(x)$$

The details of these fuzzy operators and other related information can be found in [35, 36].

There also different methods for combining each of the individual rules. Some example methods used to accumulate the results are shown in Table 3.1. The maximum algorithm is generally used for this purpose.

Table 3.1 Example accumulation methods

Method	Formula
Maximum	$\text{Max} \{ \varphi A(x), \varphi B(x) \}$
Bounded Sum	$\text{Min} \{ 1, (\varphi A(x) + \varphi B(x)) \}$
Normalized Sum	$(\varphi A(x) + \varphi B(x)) / \text{Max} \{ 1, \max \{ \varphi A(x'), \varphi B(x') \} \}$

3.3 Defuzzification Step

As the fuzzy processing component completes the rule processing and obtains an outcome, the defuzzification process begins. The main responsibility of the defuzzification step is to convert the obtained output into real output data or an action. In other words, the input of this component is a fuzzy set (aggregated output fuzzy set), and the output of the defuzzification process is a single real-life data value. Defuzzification is done according to the membership function of the output variable.

There are many defuzzification methods that have been proposed in literature all of which are based on mathematical algorithms such as centroid, maximum decomposition, and bisector of the area [39]. The two most common defuzzification methods are maximum value and centroid. The maximum value method obtains its final output value on the rule output with the highest membership function grade. This method is usually used with discrete output membership functions. The second one, centroid method, is simply the weighted average of the output membership function. It mathematically obtains the center of mass/gravity of the triggered output membership functions. Figure 3-12 illustrates the centroid calculation for an example FO calculation.

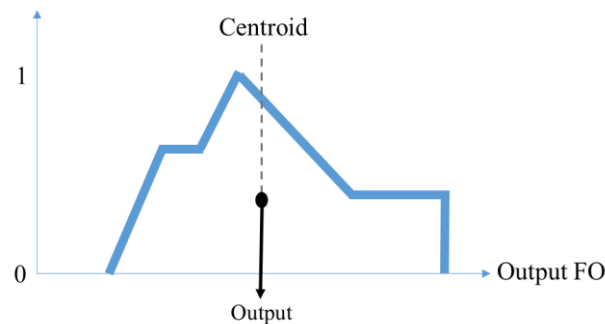


Figure 3-12 An example centroid calculation of an output membership function

From mathematical perspective, a centroid is the point in a geometrical figure whose coordinates equal the average of all the other points that make up the figure. As mentioned above, it is the center of gravity of the figure. The center of gravity for obtained FO is the output data value as shown on the X-axis in figure above that divides the area under the fuzzy membership function curve into two equal parts. This method is the most commonly used defuzzification method because it provides more precise result based on the weighted values of several output membership functions. The output value that is sent to the output interface module is the output data value at the intersection of the horizontal axis and the centroid as illustrated above.

Centroid (\bar{Y}) is calculated as in equation (1) for continuous output fuzzy sets, and with summations rather than integration for discretized variables in equation (2) where μ represents membership function output and y represent the grades.

$$\bar{Y}(\text{Centroid}) = \frac{\int_b^a y\mu(y)dy}{\int_b^a \mu(y)dy} \quad (1)$$

$$\bar{Y}(\text{Centroid}) = \frac{\sum_{i=1}^n yi\mu(y)dy}{\sum_{i=1}^n \mu(y)dy} \quad (2)$$

3.4 An Application Example of Fuzzy Logic

Now a FO calculation is described through an example with corresponding membership functions, rules and charts. The generic labels will be used for membership function labeling. This labeling is used as basis for other membership functions. These labels span from the data range's minimum point (negative large) to its maximum point (positive large). The labels are;

- NL (negative large)
- NM (negative medium)
- NS (negative small)
- ZR (zero)
- PS (positive small)
- PM (positive medium)
- PL (positive large)

These input membership functions are shown as in Figure 3-13. One input will be used for this example.

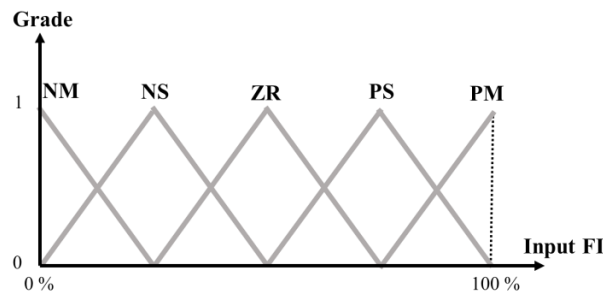


Figure 3-13 The input membership functions

The rule base that will be used for this example is given in Figure 3-14. The FI represents function input and FO represents the function output.

Rule 1)	IF <i>FI</i> is NM THEN <i>FO</i> is NL
Rule 2)	IF <i>FI</i> is NS THEN <i>FO</i> is ZR
Rule 3)	IF <i>FI</i> is ZR THEN <i>FO</i> is ZR
Rule 4)	IF <i>FI</i> is PS THEN <i>FO</i> is PL
Rule 5)	IF <i>FI</i> is PM THEN <i>FO</i> is PL

Figure 3-14 The rule base

The corresponding output membership function with respect to given rule base is shown in Figure 3-15.

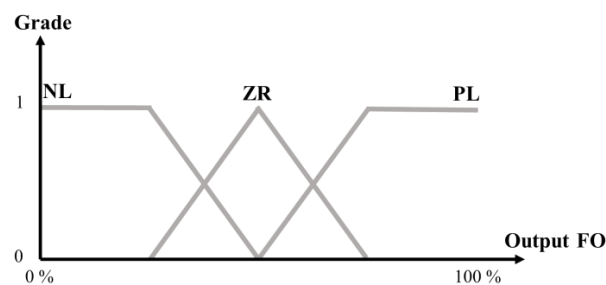


Figure 3-15 The output membership functions

As mentioned in fuzzy processing step, the FO is composed of one or more membership functions with corresponding grades. The output membership function grade is directly affected by the grade level of given input data through input membership functions.

In this example, the fuzzy input (FI) value, 60%, is provided as FI. This input belongs to ZR and PS membership functions as illustrated in Figure 3-16. Here, ZR function has 0.6 and PS function has 0.4 grades. For this example, ZR and PS membership functions will affect the output. These are determined through finding out the intersecting points for given input value. On the other hand, the output membership

functions which are chosen for output value depends on the triggered rules that are defined in rule base.

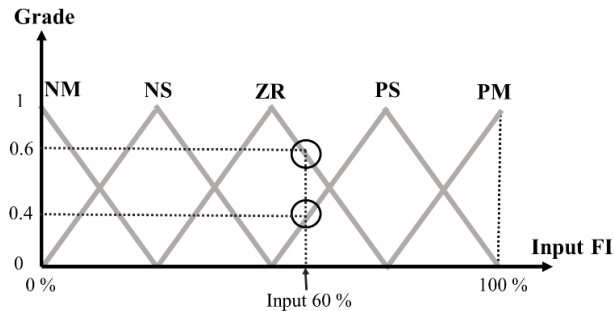


Figure 3-16 The input value and corresponding activated membership functions

For given input, the rule 3 and 4 are being triggered, because the FI belongs to both ZR and PS. As a result of this, both FO action ZR and PL should be applied to the process according to grades generated in the input membership functions which are 0.6 and 0.4 correspondingly. As a result of this employment, these 0.6 and 0.4 values are applied to ZR and PL correspondingly. Figure 3-17 shows this outputs.

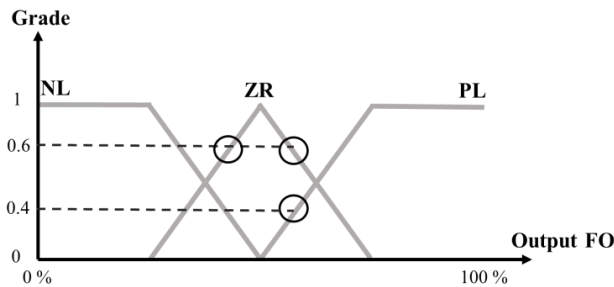


Figure 3-17 The corresponding fuzzy output values

The individual output curves corresponds to given inputs are given in Figure 3-18.

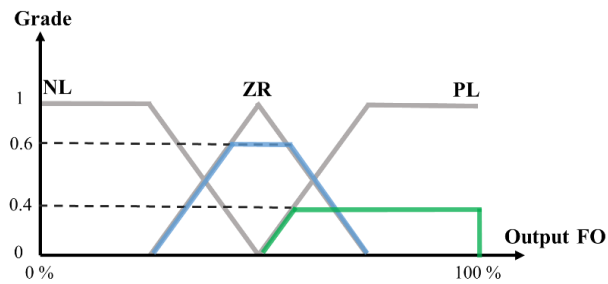


Figure 3-18 The individual fuzzy output curves

To obtain final outcome value, the fuzzy outcomes are being added together to produce aggregated outcome as illustrated in Figure 3-19.

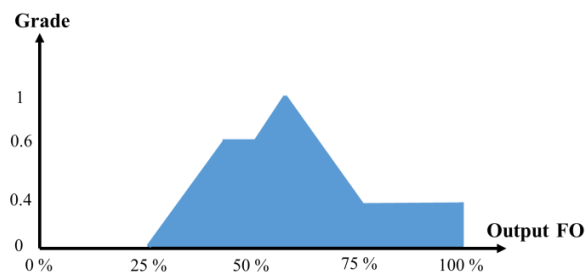


Figure 3-19 The aggregated fuzzy output curve

The final output which is generated through defuzzification step is obtained by using centroid method. For centroid method, the area of overall output curve is taken into consideration and through using the formula given in previous section, the centroid value is calculated as 65.3 and illustrated in Figure 3-20.

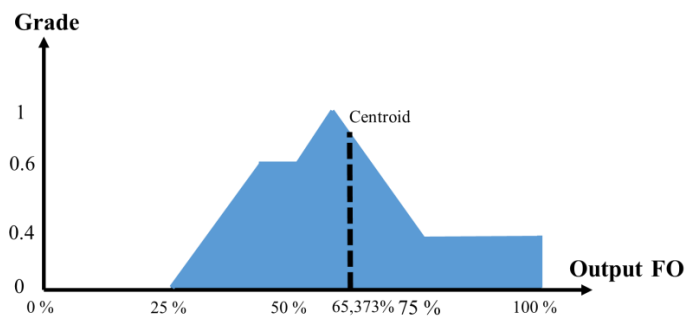


Figure 3-20 The calculated centroid value

CHAPTER 4

ENSEMBLE ADAPTIVE PREFETCHING

In this chapter, the proposed Ensemble Adaptive Prefetching (EAP) approach and its important concepts are described through given sub-sections. First of all, the architecture of EAP is depicted. Then, criterion term which is proposed with EAP is described, then the forms of data that are used in prefetching, prefetching steps and process are examined. The adaptive part of EAP is given in Fuzzy Logic-Based Adaptive Weight balancer section. All the related sections are described through referring related parts of architecture.

4.1 EAP Architecture

In this section, the proposed EAP architecture is examined in details. The proposed architecture is depicted in Figure 4-1. This architecture shows important actors with corresponding actions, relations and items. The technical details of this architecture are described in Appendix B.

The modules are illustrated through boxes and arrows show the direction of interaction. Important action descriptions are shown on these arrows with corresponding numbers which are then used to refer corresponding actions from text. The outer rectangle represents the boundary of EAP architecture. This architecture is implemented through a framework which let applications incorporate the proposed capabilities and use them through provided interfaces. The framework is simply responsible of management of criteria, requests initiated by these criteria, tile loading, and prefetching and cache replacement.

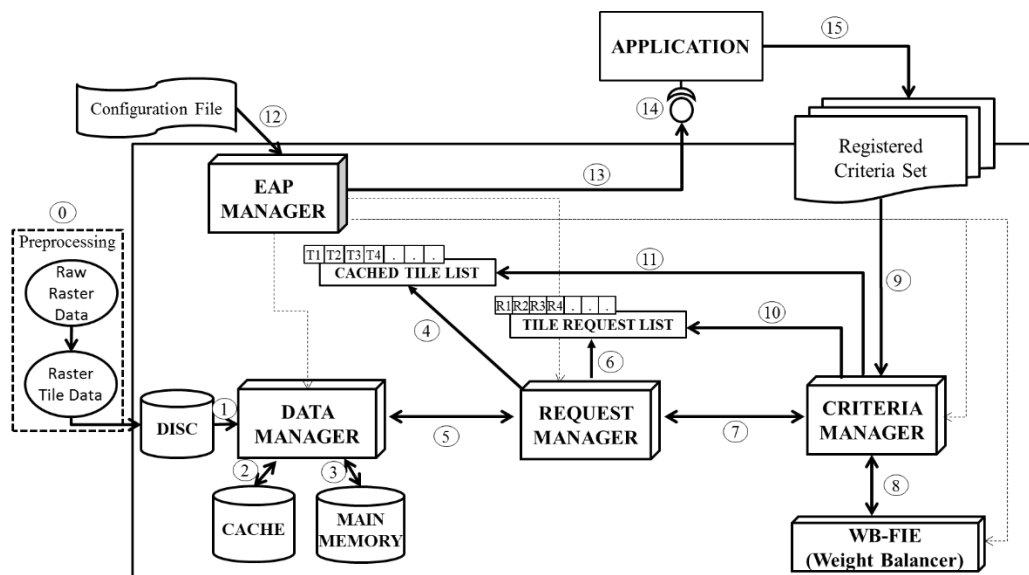


Figure 4-1 The EAP architecture

The application is abstracted from this framework through provided interface (shown with 14 in Figure 4-1). This framework can be considered as a component or library that can be integrated into any application. The main responsibility of application is to register the set of criteria or single criterion according to its requirements. The application registers set of criteria (shown with 15 in Figure 4-1) through this interface to determine the capabilities that will be used for prefetching and related operations at run-time. Then it should provide necessary criterion parameters and application state data required such as current geographic location, altitude, speed, view direction, view angle, etc. through this interface.

Finally, EAP accepts a configuration file (shown with 12 in Figure 4-1) which set initial configuration and settings for behavior of overall system. It contains parameters for data that is going to be used. It also sets up initial weight balancer parameters and rule input values. All thresholds, initial weights and other criterion specific parameters are also provided to EAP through this file. This file also contains module specific parameters and these parameters are passed to each module during initialization. The details and an example configuration file are given in Appendix B.

As illustrated in Figure 4-1 EAP has five modules which are EAP, Criteria, Request, Data managers, and Weight Balancer Fuzzy Inference Engine (WB-FIE). Each module performs the assigned task asynchronously in their own thread.

The EAP manager module is responsible for the administration of overall system and other managers. Initialization of framework through provided configuration file and interaction between application and framework is performed by this module. It provides corresponding initial configuration parameters to each module in initialization. This module obtains application input (action or data) and passes on this to responsible module. It is also responsible from monitoring execution of framework and gathering necessary statistics from other managers.

The criteria manager module is responsible for management of registered criteria. It executes each criterion operations according to initial configuration file and current state of application (shown with 9 in Figure 4-1). These operations are described in section 4.2. It passes the load requests to active criteria, obtains request priorities from each of them and provides this information to request management module to determine the load order (shown with 7, 10 in Figure 4-1). The same operation is also performed for the tiles in the cache to determine replacement order of the tiles that will be disposed (shown with 11 in Figure 4-1). It also updates criteria weights according to feedback received from weight balancer. The weight balancer uses the metrics such as criterion correction level and activity level which are given in Section 4.5 about how each criterion performs and these metrics are collected by criterion manager (shown with 8 in Figure 4-1) through employed statistics module.

The request manager is responsible for managing all load requests received from each criterion, gathering them together and determining the load order with aid of criterion manager and weight balancer (shown with 6, 7 in Figure 4-1). Then passes on these requests to data management module (shown with 5 in Figure 4-1). Holding current tile requests in a list and maintaining the order of this list is also carried out by this manager. Besides, it keeps track of tile request history and manages the state of each

request. For instance, the request initiator criteria, the current state of request whether it is loaded, or still being loaded or cached or disposed. It informs each criterion about status of its tile request according to responses received from data manager module. The conflicting situations like initiation of same tile requests from multiple criteria are resolved by this module. This module is also responsible for management of cached tiles and their replacement order as in load order (shown with 4 in Figure 4-1).

The data manager module is responsible for loading tile data from disc and management of data in main memory and cache (shown with 1, 2, and 3 in Figure 4-1). The loading operations are initiated by the requests received from request manager module in the given load order and whenever load operation is completed the request manager module is informed about completion (shown with 5 in Figure 4-1). This module also identifies and provides information about the source of data, its unique tile identifier and the other details according to provided geographic position and raster metadata given in initial configuration file. It performs simultaneous load operations according to available system resources and manages their status. The cache, its size and current cached items are also managed by this module. This module provides the loaded tile data to active criteria and application. Moving tiles from cache to main memory and vice versa is performed by this module.

The last module is Weight Balancer Fuzzy Inference Engine (WB-FIE) which is called weight balancer in short. As a result of employing multiple criteria at the same time and the versatile exhibition of application capabilities at run-time requires employment of criteria weights. These weights show the contribution of each criterion. This module is responsible from determining these criterion weights adaptively according to obtained metrics from criterion manager module (shown with 8 in Figure 4-1). To do so, it uses a rule base and fuzzy logic engine. The details of this engine, WB-FIE, and rule base are given in section 4.5.

4.2 Criterion Term

The criterion is defined as an agent that is responsible for the execution of requests, according to domain specific requirements and the current state of the application. In fact, each different prefetching approach, mentioned in related work section, can be considered as individual criterion from an EAP perspective in such a way that each of them is specifically developed to be used for given purpose. However, different from those approaches, EAP can employ more than one and different criteria together at the same time.

Criterion word is deliberately selected to describe prefetching techniques' preference on tile data. For instance, a visibility-based criterion focuses more on tiles that are in the Field of View (FOV) and prefers to load those closest to the observer before distant ones. On the other hand, a vertical cross section analysis focuses more on tiles that lie through view direction rather than whether they lie inside the FOV or not. Hence, it is believed that ensemble usage of this group of criteria help to achieve high performance and satisfy various capabilities that are used for retrieving raster data tiles. It can also be said that the more criteria application employs, the more accurate and effective prefetching it can achieve through using EAP.

It is also possible to categorize these criteria. When current literature is scanned, each individual criterion can be categorized as deterministic, adaptive or heuristic according the requirements imposed by application or developer desire. They can also be categorized as online and offline according to input being used for prefetching decision. In other words, if a criterion makes use of existing or pre-collected data then it can be categorized as off-line, however, if it chooses tiles to prefetch solely based on current state of application then it can be categorized as online. From EAP perspective, all these kinds of criterion can be used with EAP. In fact EAP, itself, is adaptive and online according to mentioned categorization.

In EAP, criterion not only covers prefetching operation. Prefetching is usually employed with other operations and usually does not operate by itself. Although some of these operations are explicitly mentioned in literature like cache replacement [8], no other operation is mentioned. Two additional operations are defined which are evaluation and requesting. All these operations are gathered under four groups: requesting, prefetching, evaluation, and replacement.

The *requesting* operation is responsible for determination of necessary geographic tiles required by criterion specific capabilities such as 2D map display or 3D terrain visualization. The main responsibility of requesting operation is to make sure that necessary tiles for given criterion is being loaded into memory. This operation makes use of direct-load requests to load tiles into memory which is going to be described in Prefetching Steps. For instance, a 3D FOV criterion request operation determines tiles that are within the FOV and initiates necessary actions to load those tiles into memory. On the other hand, 2D Map View criterion request operation determines the tiles that lie inside the map extent.

The second operation, *prefetching*, refers to abilities necessary to determine tiles that will be prefetched. In fact, this is the operation that defines prefetching related tasks and the success of prefetching technique. Here, the criterion tries to predict next possible behavior according to current state of application and then loads tiles into cache even though they are not required at that time. This operation makes use of prefetching-load request to load tiles into memory which is going to be described in Prefetching Steps. For instance, a 2D map view criterion prefetches neighboring tiles according to previous user navigation, even if they are not required or requested at that time. A 3D FOV criterion may use turning direction to prefetch tiles that will be soon visible by current FOV.

The *replacement* operation is primarily responsible for identifying the tiles that are not needed any more and need to be disposed. The tile replacement policies are employed within this operation to prevent tile thrashing or similar issues.

The *evaluation* operation takes all existing load requests initiated by other criteria and prioritizes them according to developed criterion prioritization policy. This operation evaluates existing requests initiated by other criteria and provides a preference for them. For instance, a FOV criterion may evaluate the load requests initiated by a 2D map view criterion and assign higher priorities to those that also lie inside the FOV. The other three operations use this to determine tile load, prefetch or replacement order accordingly. In fact, with this operation, different criteria specify their preferences on the tiles to find out the most required tile.

Criteria may either perform all of these operations or only some of them may be exhibited. In other words, one criterion might be used only for evaluation or for requesting, evaluating, and prefetching tiles. For instance, a criterion that takes current requests and prioritizes them according to their access frequency only needs to perform evaluation operation. It does not initiate any load or prefetching requests.

One important question about these operations is how they are triggered. There are some attempts to determine such triggers under a prefetching scheme [40]. This study identified two classes of schema: spatial and temporal threshold. In a spatial schema, prefetching occurs whenever a given pre-assigned spatial threshold is fulfilled. For instance, it is triggered when the FOV is rotated by a given angle. A temporal schema performs prefetching in a regular fashion such as prefetching once every 1 s. In EAP, threshold values are used for this purpose. Different from [40], threshold values are controlled by the criterion. These can also be changed dynamically, according to the state of the application. For instance, if the user rotates the FOV faster, then the prefetching bearing threshold value decreases. These threshold values are used for both request and prefetching operations.

In EAP, it is also possible to trigger and configure the amount of prefetching operations according to current CPU load of system. Whenever system is in idle mode or has time to perform prefetching, those are executed and load operations are

completed. This mechanism provides more flexible and configurable trigger schemas than previously mentioned studies.

4.3 Forms of Data

The data that is used in this approach is geographic. As described previously, goal of prefetching is to keep necessary subset of this data set in main memory at any given time and the rest of the data may reside in the secondary storage drive or on a network server. Moreover, considering the size of globe data, even the secondary storage of local computer might not be enough for holding all these terabyte level data. Geographic data is in the middle of this problem, labeling it might help to express and understand the overall prefetching process with its corresponding usage. From this point, these labels of data will be used correspondingly in the text to address the form of data in process. Geographic data is categorized as raster and vector data, but this study concentrates on raster data. The forms of data are given in Figure 4-2.



Figure 4-2 The forms of data

The first form of data is called *raw-data*. This covers the initial data, which is usually obtained from acquisition devices. Although this can be used directly, it is more convenient to process and transform it into a more efficient and usable format, called *raster tile data*. This data usually incorporated with a metadata (either as a separate index file or in the same file) and exhibits a different format for specific purposes (e.g. in CADRG or zip format). The raster tile data usually resides at GIS data servers or secondary devices of client computers.

Whenever raster tile data is loaded into main memory and is ready to be used by applications, it becomes either *active* or *passive data*, according to its current usage. If it is currently being used by the application, it is active; however, it is called passive

data if it is not used by the application. Passive data is either loaded through prefetching to become active later, or it comes from active data that is not being used anymore, and resides in cache. The active data is the primary data that is used by applications. However, passive data is either loaded into memory (or secondary disc) through prefetching to become active very soon or it comes from active-data that is not being used nor needed any more. Both active and passive data share same format and characteristics so that conversion from one to other requires no processing or calculation.

The last form of data is *replaced data*. This data form is introduced to identify the data that is disposed from the cache and no longer resides on memory. Although passive data can be used for data items that are not active any more, this data form introduced to identify the data that is replaced from cache and become active previously which is also being used for experimentation. Some approaches may employ specific operations for this form of data or just unload it from memory.

Usually, the size of *active/passive data* is lesser than *raster tile data*, whereas the size of raw data usually is less than *raster tile data*.

4.4 Prefetching Steps and Prioritization

After having defined the forms of data, now the overall prefetching process steps with how they make use of this data are described. Important steps of overall prefetching process with corresponding forms of data are given in Figure 4-3.

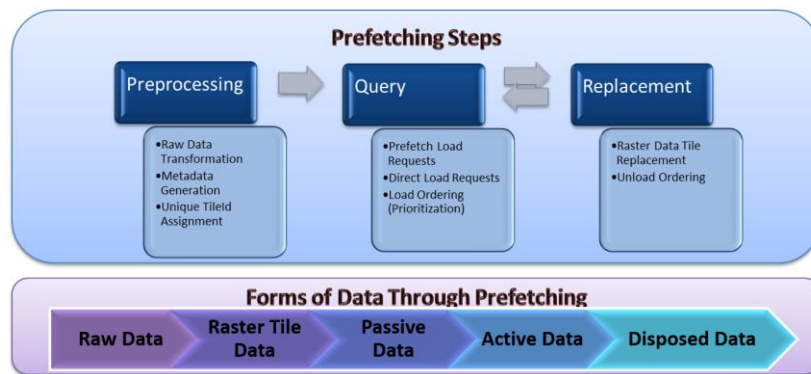


Figure 4-3 Prefetching steps and forms of data through this process

4.4.1 Preprocessing Step

The first step in prefetching is *preprocessing*, which is commonly used by many data-driven applications, but it has become more important as the scale of current geographic data is taken into consideration. The *preprocessing* covers the activities that are performed prior to execution of application. In other words, it contains any type of processing or activity that is performed on raw data which is usually acquainted with acquisition devices to prepare it for primary usage. Usually, this preliminary data is transformed into a format through some data mining practices and data preprocessing algorithms that will be more easily and effectively processed for the prefetching. Another important motivation of preprocessing is alleviate the burden of executable models and provide more effective and interactive execution, query and data retrieval operations at execution time.

The preprocessing is also used for incomplete data that may come from “Not applicable” data, human/hardware/software problems or noisy data especially for image or signal data that may stem from faulty data collection instruments, human or computer error at data entry or errors in data transmission. Finally, it is used for inconsistent data that comes from different data sources.

At this step, according to nature of data, compression, identification and encoding operations can be applied to raw data. Moreover, operations necessary for tiling are usually applied at this stage.

For EAP, raster data tile files are generated from raw data, according to developed unique raster tile ID determination. These files are being produced according to developed unique raster tile id determination and then a metadata index file that contains information depicted by Figure 4-4 about the preprocessed data is generated.

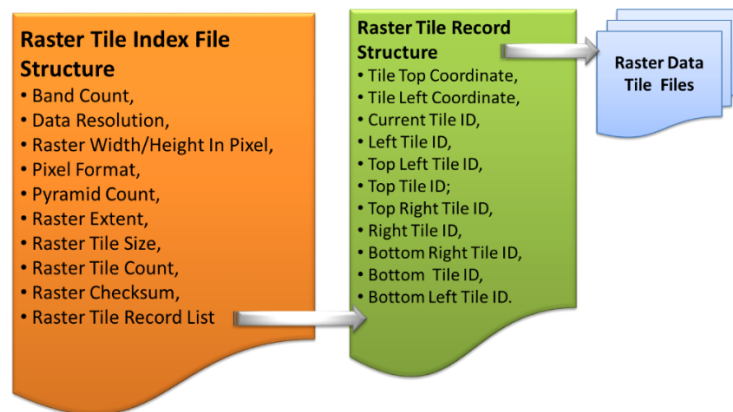


Figure 4-4 Raster Metadata index file structure

These raster tiles are assigned unique identifier (ID)s according to their resolution, location, and origin. With such assignment, candidate tile search is no longer required because a simple calculation to determine which tile search is to be retrieved at runtime is made, according to its geographic location. The raster tile data files are generated in tiles which are addressed through mentioned metadata files.

Another contribution that has been done in this phase is to assign these raster tiles unique ids according to their resolution which is called unique raster tile id determination. The acquired geographic data refers to unique locations on earth, from this fact, this data is partitioned into sub-partitions and enumerate them according to resolution or map scale. As a result of this very simple technique, the search of tile to load is not required anymore. To find out which tile is required to be retrieved at

runtime is nothing but a simple calculation. In fact, raster data files that are generated in preprocessing phase are named according to this id. These IDs are stated with unsigned integers which provide $<0, 4.294.967.295>$ range.

4.4.2 Query Step

In this step, all prefetching and loading of candidate tiles occurs, as well as querying raster tile data, according to the layout determined in the previous step. According to nature of prefetching, it might be complex or as simple as loading raster tiles directly into memory. It involves querying the raster tile data whose layout is specifically designed so that queries can efficiently identify this data at runtime. This is provided through the unique id assignment defined in previous step where query operations become the calculation of this unique id.

The loading of tile data is performed through two operations: *direct* and *prefetching load requests*. The *direct load request* usually occurs at the beginning of application execution where no information for prefetching exists, or when drastic changes occur in geographic locations. Moreover, these are initiated when the prefetching technique fails to predict the corresponding raster tiles. These are the situations where there is little or no information for prefetching exist and application required to use raster tile data that is not exist in cache or loaded into main memory yet.

Whenever sufficient information for prefetching is obtained, the *prefetching load requests* are initiated and the data that will be probably required by application is loaded into cache. This is generated according to the nature of individual criteria. Whenever the application requires this data, rather than loading corresponding data, the already loaded passive data becomes active and is provided to the application. Later whenever application requires this data, instead of going to secondary disc or server and load corresponding data, the passive data becomes active and provided to application.

The data that stays on secondary disc or server is in not loaded state. Whenever it is loaded into memory and become an active tile, than it pass to active state and whenever it is moved into cache it pass to cached state. Finally, it passes to disposed state whenever it is unloaded from memory. These states closely resemble the forms of data mentioned.

An important element of the query step, prioritization, is described in the following section.

4.4.3 Prioritization

An important issue independent of the origin of request (prefetching or direct load request) is load ordering. Certain prefetching mechanisms focus on load ordering rather than determining candidate tiles. The most straightforward approach is to load these items in the order in which they were requested. However, this approach could be problematic when user moves rapidly or multiple cache misses occurs, so the item in the load-order may lose its importance or become irrelevant. The RAP [4] method follows this approach, and the load order is predetermined according to movement direction. In this case, each neighboring tile is represented by a number and a fixed order of these tiles are loaded according to movement direction whenever a transition occurs. This numbering and load order is shown in Figure 4-5.

North			DIRECTIONS	TILE ORDER	DIRECTIONS	TILE ORDER	
West	0	1	2	EAST	3, 2, 4	NORTHWEST	0, 1, 7, 2, 6
	7		3	WEST	7, 6, 0	NORTHEAST	2, 3, 1, 4, 0
	6	5	4	NORTH	1, 0, 2	SOUTHWEST	6, 7, 5, 0, 4
South			SOUTH	5, 4, 6	SOUTHEAST	4, 5, 3, 6, 2	

Figure 4-5 a) The tile numbering, b) load-order for each direction in RAP according to (a)

Although this approach is sufficient in RAP cases where the scope is simple 2D

navigation, it becomes more complicated when 3D and other criteria are involved, and number of tiles that are going to be fetched increased. In fact, it is highly possible that some individual tiles are required before than others in 3D. Nevertheless, the order of required tiles might be changed dynamically, according to 2D/3D.

It is critical to reduce the user perceived delays which becomes an important motivation of our EAP approach. Although the misses cannot be prevented completely, necessary actions could be initiated to reduce this delay. This is achieved with dynamic load order and loading the nearest most important tiles before then others. Hence, first the predetermination of order is abandoned which does not provide best results in most of the common scenarios and instead incorporate a prioritization policy. These requests are evaluated and assigned priorities to those requests such that the higher prioritized tile is loaded before other ones according to criterion specific conditions and requirements.

In EAP, this ordering is performed by employing mentioned raster tile prioritization. As a result of using an ensemble of prefetching techniques, EAP provides more precise load ordering than individual prefetching mechanisms. This is very important for prefetching, as it helps to access those tiles that are more likely to be required before others to prevent possible misses. Moreover, application delays and lags are reduced for *direct load requests*. Hence, prioritization policy mechanisms are incorporated into EAP through each criterion where each request is evaluated and assigned priorities by each prefetching technique, and higher prioritized tiles are loaded before others.

Moreover, this same policy is also used for tile disposal in such a way that this time the lower prioritized tiles are being selected for disposal. As a result of using same mechanism, no extra cost or mechanisms are required for disposal and priority policy developed for prefetching can easily be used for disposal.

For *direct load request* and *prefetching load request*, 1.0–100.0 and 0.0–1.0 ranges

are used, respectively, to make application load required tiles before prefetched ones. These ranges ensure that the tiles that would be required directly by application are assigned higher priority values than prefetching candidate tiles. It should also be noted that all these range values and parameters could easily be changed through configuration files.

The prioritization policies are dependent on each individual prefetching techniques; thus, three policies are developed for three categories of prefetching approaches, which are 2D prefetching, 3D prefetching, and analysis prefetching.

The prioritization policy used for 2D prefetching in EAP is illustrated in Figure 4-6. Here, the map extent represents the portion of map tiles (which is 4 x 3) that the user sees in 2D navigation mode in her screen. Most of the 2D map visualization applications follow this representation where number of map tiles are being displayed inside a window to user and user navigate on this map through well-defined operations like pan and zoom. User expect to see flawless map transition as she pan and navigate on map. So the policy should take this into consideration and employ prefetching according to this motivation. It is believed that the policy proposed for 2D map navigation can be used for most of the 2D map visualization and navigation prefetching operations.

Raster tiles that are within the map extent are assigned priorities from 1.0 to 100.0, according to the geographic distance of their center to the map extent center. In other words, tile that contains the map extent center has the highest priority value which is 100.0 and the tile that lies at the boundary of map extent has 1.0 priority value. The tiles that lies among map extent center and boundary are assigned priority values linearly according to their distances.

The tiles that are outside the map extent are assigned priorities from 1.0 to 0.0 up to a maximum range. In other word, tiles that are just outside the map extent are assigned priority value 1.0 and this priority decreases as tiles get further away from the center

of map extent. The priority assignment is limited with a range in such a way that if a tile is farther than this given max range it is assigned priority value 0.

In addition to this distance based prioritization, this policy make use of navigation direction for priority assignment. After obtaining distance-based priority, the tiles are evaluated according to the next possible movement direction. This direction is determined by considering previous user navigations and if a tile is in the possible movement direction of the map extent, then its priority is increased by an extra 25%.

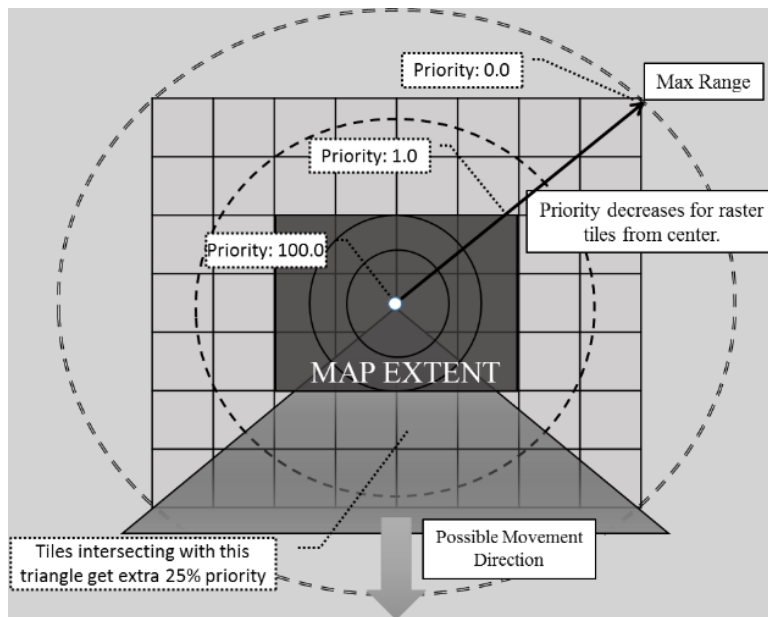


Figure 4-6 2D Navigation prioritization policy

The prioritization policy for 3D prefetching is illustrated in Figure 4-7 which is primarily developed for 3D visualization and navigation applications. Here, FOV triangle represents the user's or camera's current field of view where all 3D visualization based on. The most of the 3D visualization applications like 3D simulations or fly-through applications follow this representation where a 3D world or terrain is being constructed according to current FOV. This world or terrain is usually generated through finding the FOV intersected elevation and map tiles. However, in 3D, the user navigation is more complex than the 2D case where user

can rotate its FOV or change its angle or change the other view parameters like heading and pitch. In addition to these, she may ascend or descend which changes the number of raster tiles required to be fetched. Moreover, different kind of combined navigations is possible such as moving one direction as looking backwards or other sides. Such issues are taken into consideration while designing the 3D FOV prioritization policy to serve these kinds of applications.

Raster tiles that are within the FOV are prioritized according to their distance from the navigator, as in the 2D case from 1.0 to 100 according to the geographic distance of their center to eye point/navigator location. In other words, tile that is just in front of navigator view or contains navigator has the highest priority value which is 100.0 and the tiles that lie at the boundary of FOV triangle has 1.0 priority value. The tiles that lie among navigator location and FOV triangle boundaries are assigned priority values linearly according to their distances as in 2D policy.

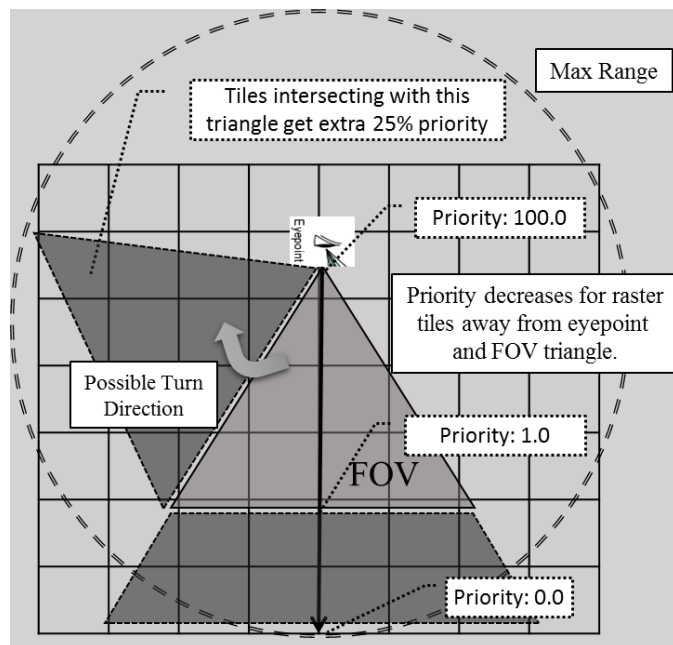


Figure 4-7 3D FOV prioritization policy

The tiles that are outside the FOV triangle are assigned priorities from 1.0 to 0.0 up to a maximum range as in 2D case. In other word, tiles that is just outside FOV

triangle and near to navigator are assigned priority value 1.0 and this priority decreases as tiles getting further away from the FOV triangle and navigator. Different from 2D map extent case, the priority values of the tiles that are outside the FOV triangle but near to navigator location assigned higher priority than the farther tiles which is illustrated in Figure 4-8. The maximum range could also be set as far clipping plane parameter of perspective projections which is used for 3D visualization. The priority assignment is limited with a range in such a way that if a tile is farther than this given max range it is assigned priority value 0.

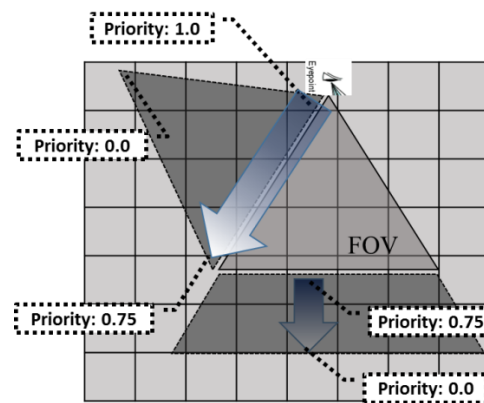


Figure 4-8 How outside tiles are being prioritized in 3D FOV policy

The remaining tiles are prioritized according to their distance from the FOV triangle up to a maximum range. Moreover, as in the 2D case, the tiles that are within the next possible turn or move direction (clockwise or counter clockwise) are given higher rates than other. For instance, if user turns its heading clockwise for given period then it probably continues turns its heading in clockwise direction. As a result of this, those tiles are assigned extra priorities as illustrated in Figure 4-7. In addition to these turn directions, if it moves in forward or backward, those tiles that lie after or behind navigator are also assigned higher percentages.

Finally, a prioritization policy for analysis prefetching in EAP is developed for linear GIS analyses like vertical cross section analysis as illustrated in Figure 4-9. Here, the line represents the line that analysis is interested. This line may move with navigator

movement or it can be applied to a given fixed geographic location. When used with navigator, its directions are changed according to view direction of navigator. Its length can also be changed. This policy mainly aims to find and prioritize tiles that intersect with this line and around it.

Raster tiles that intersect with analysis cross section are prioritized from 100.0 to 1.0, according to their geographic distance from analysis start point to end point. In other words, the tiles that contain the analysis start point are being assigned priority value 100. On the other hand, the one that contains the analysis end point is assigned priority value 1.0. The priorities of rest of the tiles that lie among these two points are assigned linearly. The tiles that intersect with the extension of the cross-section are again rated in a similar way, but with a 0.0–1.0 interval till a given maximum range.

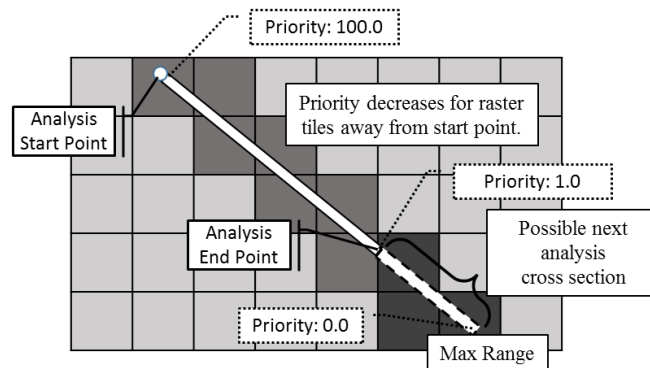


Figure 4-9 Analysis prioritization policy

The final priority of a tile is obtained by adding the weight-multiplied priority values of each prefetching policy described above. These weights, which are in the 0.0–1.0 range, determine the influence of each criterion. These grading/prioritization operations are performed whenever a load operation is initiated or policies are triggered. When a load request is initiated, currently active prefetching techniques calculate corresponding tile's priority value for the first time and place it into a priority-based load list.

The tiles are re-graded whenever prioritization policies are being triggered. For 2D

prefetching, the policy is triggered when the map extent is panned for a given distance or a zoom operation is performed. For 3D prefetching, it is triggered by FOV bearing and FOV angle changes and viewer movement for given thresholds. Finally, for analysis prefetching, it is triggered whenever the analysis start point or analysis range is changed. As these ranges and threshold values can be configured prior the execution of application, they can also be changed at run time according to current state of application. For instance, if a user moves rapidly the threshold value used for panning is increased to prevent any unnecessary raster tile loads.

It should be noted that the proposed three policies can easily be used for mentioned kinds of applications as basis. With EAP, other prioritization policies can also easily be developed and integrated into the system to increase the efficiency and precision of prefetching.

As a result of this mechanism, tile load order may change till the start of the load operation which provides great flexibility. This is also the case for cache replacement where the cached tiles are being held till the start of tile disposal which prevents any unnecessary disposing and reloading operations.

Before closing prioritization discussion, the overall process of prioritization, its usage in overall execution of EAP is going to be described in a more formal way. These can easily be correlated with previously mentioned prefetching elements.

- Let \mathcal{S} be the current state of application. This state contains any information that is used by criterion for prefetching like system mode (whether 2D or 3D visualization activated) or speed or altitude, etc,
- Let \mathcal{L} be the geographic location of navigator. This is an important actor and trigger for all prefetching operations which also represent the navigator being used for most of the prioritization policies,
- Let ϵ_i be the ***i*th** criterion that is currently used for prefetching.
 - θ_i be the weight of ***i*th** criterion. Different from other approaches, EAP can dynamically calibrate the weight of each criterion. How these weights are being calibrated is going to be described in section 4.5.
 - τ_R be the *direct request operation* threshold.

- τ_P be the *prefetch request operation* threshold. These two are the threshold values that are specific to each criterion.
- Let r be the single tile request. There is a set of priority values assigned by each criterion and an overall priority ρ_r of corresponding request,
- Let ordered set of requests be represented as S_o and unordered request set as S_u . This order is determined by using each active criterion through the evaluator operation.
- Let L_A and L_C be the current Active and Cache lists,
- Let H be the priority based ordered request list,
- Let f be the evaluator function specific to each criterion.

Now on, we can illustrate how EAP prefetching occurs in a formalized manner.

- So at any time t_0 while application is in $\$$ state;
 - If τ_P is **satisfied** for corresponding ϵ_i then a set of tile requests $S_u = \{r_1, r_2, \dots, r_n\}$ where n is the number of **prefetch tile** request initiated,
 - $\forall r$ in S_u calculate;

$$\rho_{r1} = \sum_{i=0}^n \theta_i * f(\epsilon_i, \$, L, r_i)$$

$$\rho_{r1} = \sum_{i=0}^n \theta_i * f(\epsilon_i, \$, L, r_i),$$
 where n is the no of active criteria and generate set of $S_o = \{r_i, r_{i+1}, \dots, r_n\}$ through H where request are ordered according to calculated priorities from highest to lowest and start with highest-valued request for loading.
 - Loaded tiles are put into cache tile list (passive data),
 - If τ_R is **satisfied** for corresponding ϵ_i then a set of tile requests $S_u = \{r_1, r_2, \dots, r_n\}$ where n is the number of **direct tile** load requests initiated,
 - Follow the same procedure, only loaded tiles are directly put into active tile list.
 - At any time if ϵ_i realize that a requested tile r is not needed any more by application then move corresponding item from L_A active tile list to L_C cache list.
 - At any time **if cache size is reached to a defined range** then remove the provided number of items from H that has lowest priorities to generate space for new tiles.
 - The weight of each criterion might be changed during run-time.
 - A criterion may decide to ignore a request which is initiated by other criteria according to current application state.

4.4.4 Replacement Step

Through the query step, raster data tiles are loaded into memory through either *direct* or *prefetching load requests*. To have a more efficient data management infrastructure and to prevent any unnecessary I/O operations, most prefetching systems employ a cache; thus, rather than disposing not required data from memory directly, it is first placed into cache where it can be accessed later. For EAP, there are two sources of cache data: the first is the recently used active data, which has become

passive, and the other is from the employed prefetching algorithm.

Whenever a new raster data tile is required by the application, the cache is checked first. If it is found there then a cache hit occurs; however, if it is not found there then a cache miss occurs and a *direct load request* is initiated to load data. The employed cache in the proposed approach holds processed raster tile data, and its size can be configured either before the application is executed or during run-time, according to system resources.

The *replacement step* occurs when the current cache is full and space is required either for not-required active items that will become passive or for newly prefetched items that will be placed into cache. Although removing the tiles in the order in which they are loaded is an option, it is not feasible to follow this policy. There are many well-known policies such as last recently used (LRU), least frequently used (LFU), or most recently used (MRU), which can be employed for replacement. Although these are good candidates, in EAP, a priority-based replacement policy is employed. Hence, the same mechanism that is used for load ordering can also be used for replacement, in such a way that items that have least priority are replaced first. The motivation behind this choice is similar to one that is mentioned in load order discussion, which is to be able to change the order of items till the start of dispose operation. For instance, the priority of a raster tile data that is a candidate for replacement may increase as the viewer goes closer to that tile; thus, these tiles need to be evaluated before they are directly replaced. Moreover, as a result of using the same mechanism for both loading and replacement, less processing is required for selecting candidates for disposal.

Another issue in the replacement step is when the cache is being checked for disposal. The replacement operation is triggered when current cache size is reached to a user provided or a default 75% limit. The cache size is checked whenever a prefetching operation is initiated or tiles are required to be moved from main memory to cache. If the available cache size is reached to a given limit, then a user provided or default

number of tiles are disposed according to the priority based replacement policy. The default number of tiles to dispose and cache limit is determined through following method given in [4].

4.5 Fuzzy Logic-Based Adaptive Weight Balancer

Having described the EAP process model, the adaptive part, which is based on fuzzy logic, is going to be described in detail. This is critical, especially when more than one criterion is being employed. The output of this step is criterion weight which is, in fact, being used by prioritization. Its primary purpose is to increase the overall prefetching performance (hit ratio) by continuously changing the weights of each criterion at run-time. For this purpose, the weight of a well-performing criterion is increased to make it more prevalent and that of poorly performing ones is decreased, which reduces their negative contribution to the hit ratio. Then, these weights are multiplied with tile priority values that are obtained with prioritization policies to get a final load order. Note that these weights may change according to exhibited capabilities at run-time.

The main motivation of selecting fuzzy logic for proposed approach is its suitability for prefetching problem. The navigations and capabilities that are handled by prefetching can vary from one application type to other and even in one application from start of session to end. So, it is not possible to find a statistical or mathematical model that can fit and cover all these cases. In fact, it is believed that prefetching problem itself cannot be expressed in precise numerical terms. However, describing problem with words from natural language through some well-defined rules seems more suitable for prefetching. At this point, other approaches entail accurate equations to model such real-world behaviors. On the other hand, fuzzy logic design can accommodate the ambiguities of real-world in such a way that it provides both an intuitive method for describing systems in human terms and automates the conversion of such system specifications into effective models. Hence, fuzzy logic rapidly become an important to for developing sophisticated control systems. As

described in Chapter 2 and previous paragraph, employing and managing set of prefetching techniques can be considered as a control problem and so fuzzy logic fits in with given problem.

Fuzzy logic can be described as an extension to traditional logic for dealing with the knowledge representation problem in an environment of uncertainty. It is a form of logic where underlying modes of reasoning are approximate rather than exact. In fact, fuzzy logic stems from human reasoning, which are usually based on approximate reasoning [35]. Moreover, it is described as a rule-based decision-making technique, employed for expert systems and process control, which primarily aims to emulate the heuristic, rule-of-thumb approach of human reasoning to many problems. Similarly, from a prioritization perspective, it may not be easy to label tiles as load first or last. According to overall priority of each tile, a grade might be assigned to each of them, which make fuzzy logic suitable for weight balancing. For this purpose, a weight balancer-fuzzy inference engine (WB-FIE) is developed to modify these weights. The overview of WB-FIE is shown in Figure 4-10.

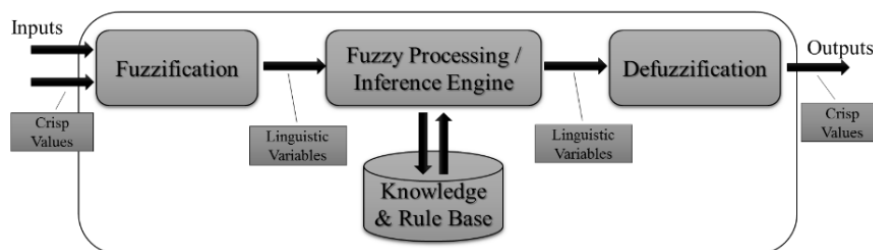


Figure 4-10 The architecture of a weight balancer-fuzzy inference system

There are three important components of WB-FIE: fuzzification, fuzzy processing, and defuzzification with input and outputs.

Fuzzification is responsible for converting input data into fuzzy membership; in other words, linguistic variables. A linguistic variable is a one that uses linguistic values such as low, medium, and high. This is required to process rules that are defined in terms of linguistic variables. These input values are analyzed according to user-

defined charts, which are called membership functions and used to group received input data into fuzzy sets. These functions are usually real-valued functions 0.0–1.0 and range outputs according to how well input fits into each function where 1 value is at the center of the given set. These functions are defined by corresponding linguistic variables, which will shortly called as label [36].

For WB-FIE, criterion activity level (CAL) and criterion correction level (CCL) are defined as input variables in term of percentages. The CAL is obtained by observing the activities of individual applications. For instance, in case of 3D applications, if FOV bearing or angle changes frequently, then CAL will increase. Similarly, in case of 2D applications, it increases if user pan continuously. The CAL decays by time if no activity occurs. If any activity occurs, then the activity level is set to 50%. The contribution of each activity to CAL is specific to each criterion. For a 3D criterion, the FOV angle, bearing, pitch, and location changes are considered as contributors. Bearing change is the highest contributor with 80% and remaining 20% shared among other activities. These percentages are obtained through observing the frequency of activities exhibited in 3D applications. The CCL value represents the correct prediction level and calculated by equation (3).

$$\text{CCL} = (\text{No. of Correct Predictions [hits]} / \text{Total No. of Predictions}) * 100 \quad (3)$$

The membership functions may have various shapes, depending on the data set such as S, Z, Λ , and π [39]. For WB-FIE, Λ -shape is selected because of its simplicity and to ensure smoother control over input. The membership functions for CAL and CCL are shown in Figure 4-11 and Figure 4-12 correspondingly.

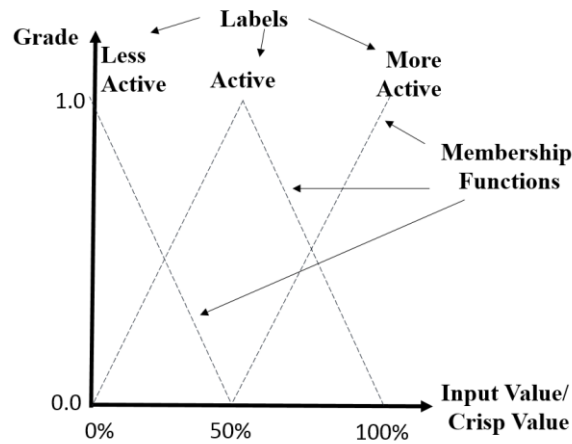


Figure 4-11 The criterion activity level membership functions

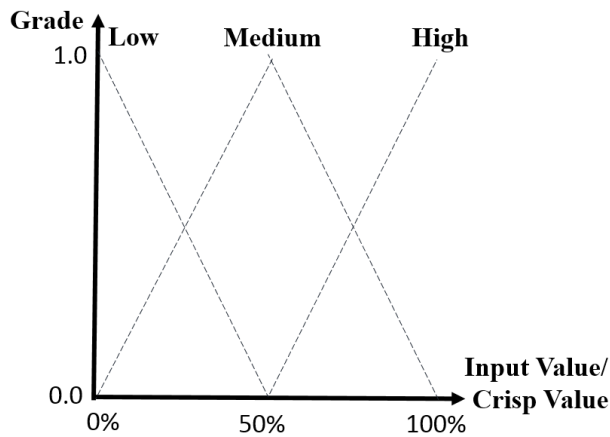


Figure 4-12 The criterion correction level membership functions

The fuzzy processing component is primarily responsible for analyzing the input data that is defined through membership functions and arrives at a control output. This is performed by mapping from input fuzzy sets into output, according to the information stored in the rule base. After the inputs are processed as previously mentioned, the corresponding input fuzzy sets are passed to this component, which processes the current inputs using the rules retrieved from the rule base. Each of these rules is in IF-THEN form such as “IF X1 is A1 and ... and Xn is An THEN Y is B.” According to activated rules, the control output is generated based on the THEN part of these rules. A rule is activated if its input conditions, i.e., the IF parts, are satisfied. There

might be cases where more than one rule is activated when all conditions are evaluated. Fuzzy operators that are used for this purpose are described in [35, 36]. The rules developed for WB-FIE are listed in Figure 4-13. The number of rules which is nine kept as small as possible to achieve a less complex inference engine. In fact, these rules perfectly cover different types of the capabilities and behaviors as illustrated in experiments conducted which are given in Chapter 6. Moreover, these can be extended or configured through again provided configuration file.

- IF CPR is **LOW** and CAL is **LESS ACTIVE**
THEN **Decrease More** the criterion weight.
- IF CPR is **LOW** and CAL is **ACTIVE**
THEN **Decrease** the criterion weight.
- IF CPR is **LOW** and CAL is **MORE ACTIVE**
THEN **Preserve** the criterion weight.
- IF CPR is **MEDIUM** and CAL is **LESS ACTIVE**
THEN **Decrease** the criterion weight.
- IF CPR is **MEDIUM** and CAL is **ACTIVE**
THEN **Preserve** the criterion weight.
- IF CPR is **MEDIUM** and CAL is **MORE ACTIVE**
THEN **Increase** the criterion weight.
- IF CPR is **HIGH** and CAL is **LESS ACTIVE**
THEN **Preserve** the criterion weight.
- IF CPR is **HIGH** and CAL is **ACTIVE**
THEN **Increase** the criterion weight.
- IF CPR is **HIGH** and CAL is **MORE ACTIVE**
THEN **Increase More** the criterion weight.

Figure 4-13 WB-FIE fuzzy logic rule list (rule base)

Fuzzy logic rules which have two inputs can also be represented in a matrix form to represent AND conditions. An advantage of this representation is that it makes it easy to represent all the rules for a system in a more compact form. These rules are displayed in Table 4.1

Table 4.1 WB-FIE fuzzy logic rule matrix (rule base)

CPR \ CAL	LESS ACTIVE	ACTIVE	MORE ACTIVE
LOW	Decrease More	Decrease	Preserve
MEDIUM	Decrease	Preserve	Increase
HIGH	Preserve	Increase	Increase More

For instance, Figure 4-15 illustrates a 3x3 matrix and 9 rules in total that uses two inputs X_1 and X_2 , and one output Y_1 .

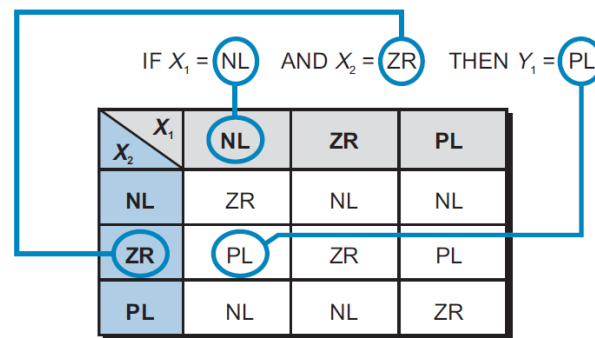


Figure 4-14 The mapping of list based rule to matrix representation

After one or more rules are activated, the outputs of each rule are combined into a single fuzzy set, which is considered as an outcome. This is the collection of one or more output membership functions, again with labels. The criterion weight operation (CWO) output membership functions are shown in Figure 4-15. Here, according to the obtained output, the multiplier for changing the weight of the criterion is provided. These operations are then analyzed and applied to each criterion, according to their current weight.

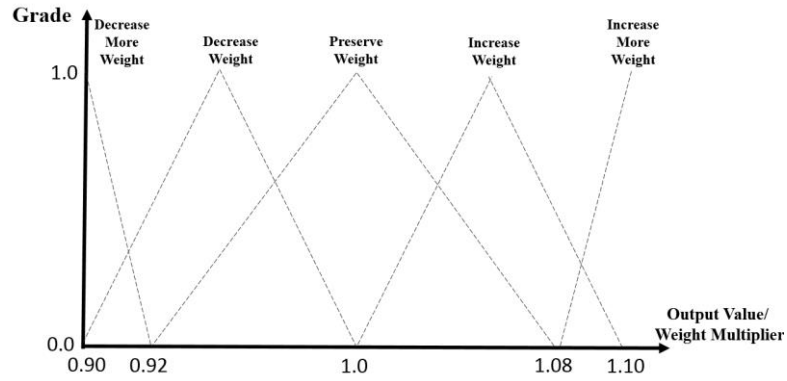


Figure 4-15 The criterion weight operation membership function

As the fuzzy processing component completes the rule processing and obtains an outcome, the defuzzification process begins. The main responsibility of the defuzzification step is to convert the obtained output into real output data or an action. In other words, the input of this component is a fuzzy set (aggregated output fuzzy set), and the output of the defuzzification process is a single number. Various defuzzification techniques have been proposed in the literature such as centroid, maximum decomposition, and bisector of the area [39]. The most common one is centroid, which is also used for WB-FIE. This method is simply the weighted average of the output membership function. The centroid method is used for WB-FIE because of its simplicity and commonality. This method applies to both non-continuous, or discrete, output membership functions, as well as continuous ones. The controller uses approximate digitized values for membership functions to compute each of the points in the summation. How this centroid is calculated is given in equation 2 at section 3.3.

The output of WB-FIE in terms of CAL and CCL is CWO, which is illustrated above. According to the obtained output, the EAP changes the weight of the corresponding criterion. For instance, if the activity level of a criterion is high with its high precision ratio, then EAP increases its weight to emphasize its influence on load order, which places this criterion's preferred tiles at the front of the load list.

4.5.1 A Sample WB-FIE Execution

In this section, an example WB-FIE execution is going to be described through the corresponding fuzzy logic components that are discussed above to illustrate the adaptive engine working logic. Same input/output membership functions and rules are going to be used for this purpose which are CCL, CAL and CWO.

The input values CCL and CAL are being collected periodically and fed into adaptive engine. So at one point, assume that the following input crisp values (32.5% for CCL and 60% CAL) are observed and fed into the WB-FIE.

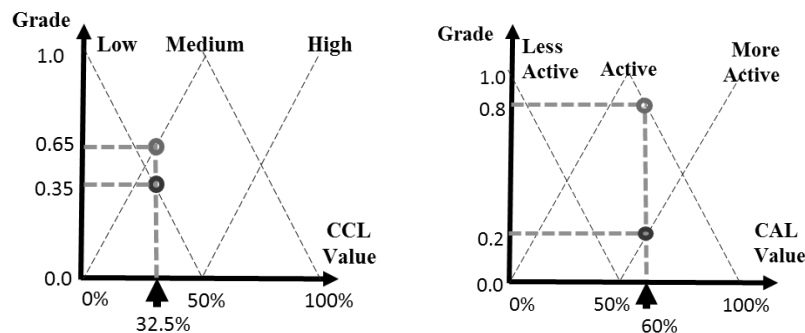


Figure 4-16 The input values and CCL/CAL membership functions

So at first stage, these crisp input values should be converted into semantic values through fuzzification. It should be noted that these two inputs have four intersections with four membership functions namely; low and medium for CCL, more active and active for CAL. These intersection points are illustrated with small circles on membership functions as illustrated in Figure 4-16. As a result, these inputs are semantically described as with these labels. Then through fuzzy processing steps, these converted inputs are used to find triggered rules and then overall outcome should be calculated. According to rules that are given in Figure 4-13 and Table 4.1, four rules are being triggered as illustrated below in Figure 4-17. Each output of triggered rule is depicted through drawn circles. Now, overall process is going to be described through these triggered rules.

CCL \ CAL	LESS ACTIVE	ACTIVE	MORE ACTIVE
LOW	Decrease More	Decrease	Preserve
MEDIUM	Decrease	Preserve	Increase
HIGH	Preserve	Increase	Increase More

Figure 4-17 The four triggered rules are shown in circles

The inputs for the first triggered rule are CCL=Low and CAL=MoreActive and these two inputs maps to output CWO=Preserve. The corresponding mappings and mappings of inputs on output membership functions are shown in Figure 4-18. As illustrated in Figure 4-18, the output is illustrated as areas as a result of having continuous membership functions.

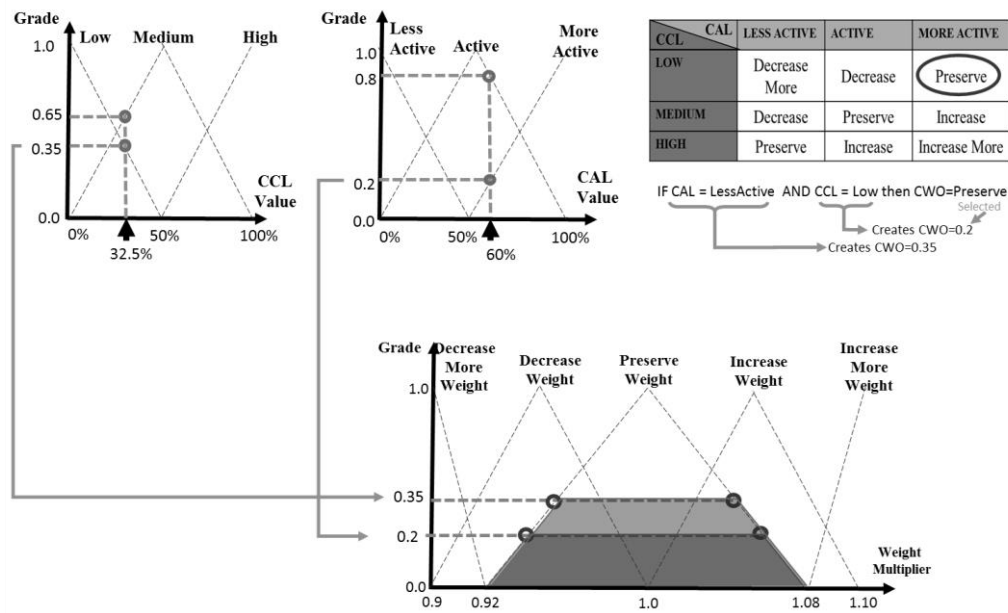


Figure 4-18 The operations occurred at fuzzy processing step for given triggered rule with corresponding mappings

Then the smaller output is being selected among these two outputs as a result of triggered rule. The rule triggered has `AND` fuzzy logic operator and as mentioned before fuzzy logic and operator takes smaller input as illustrated in Figure 4-19.

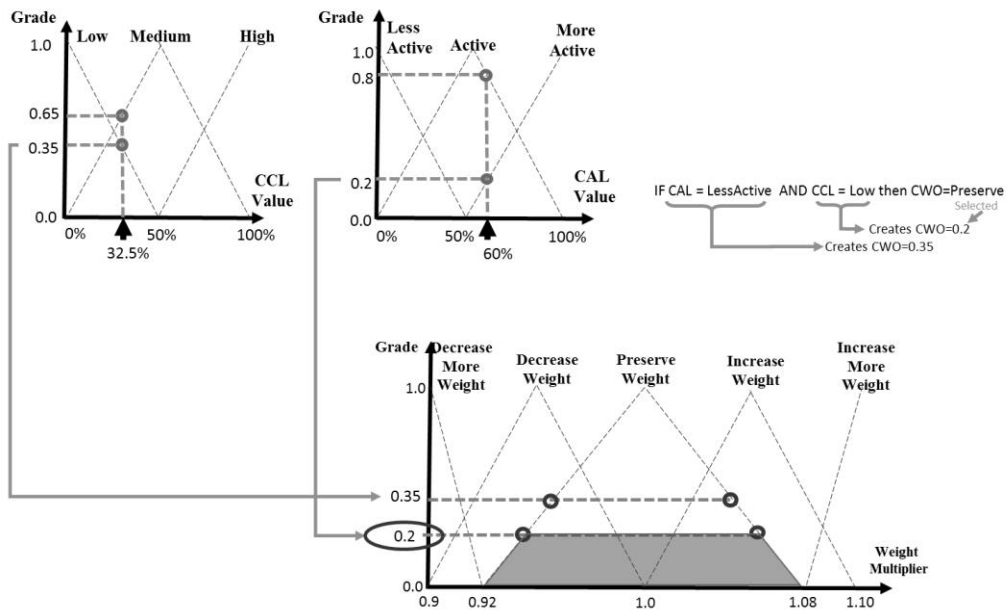


Figure 4-19 The selected output for triggered rule

Similarly, the other triggered three rules in fuzzy processing stage are illustrated in Figure 4-20, Figure 4-21, and Figure 4-22. The arrows are originated from the inputs used in triggered rule. The selected outputs and intersected membership function are illustrated through circles.

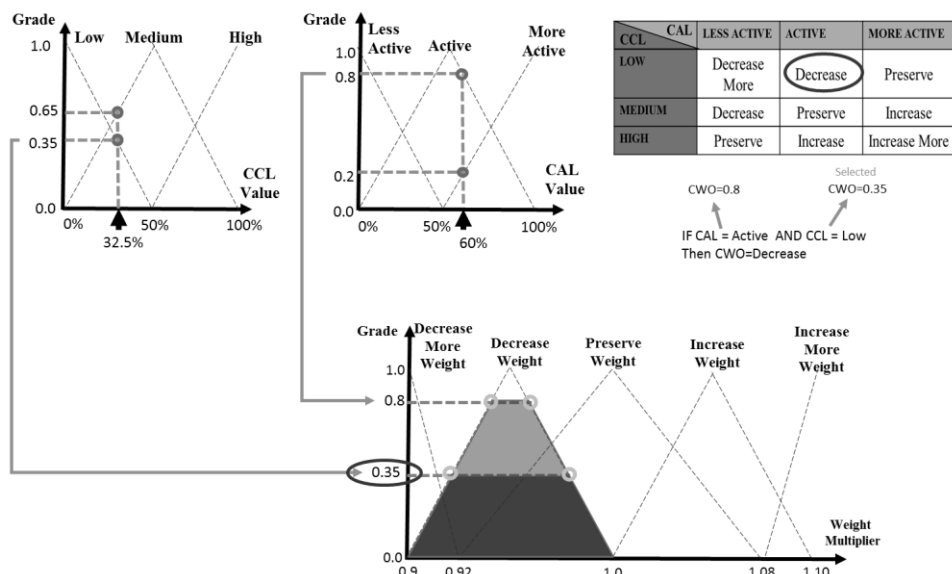


Figure 4-20 The second triggered rule with corresponding mappings

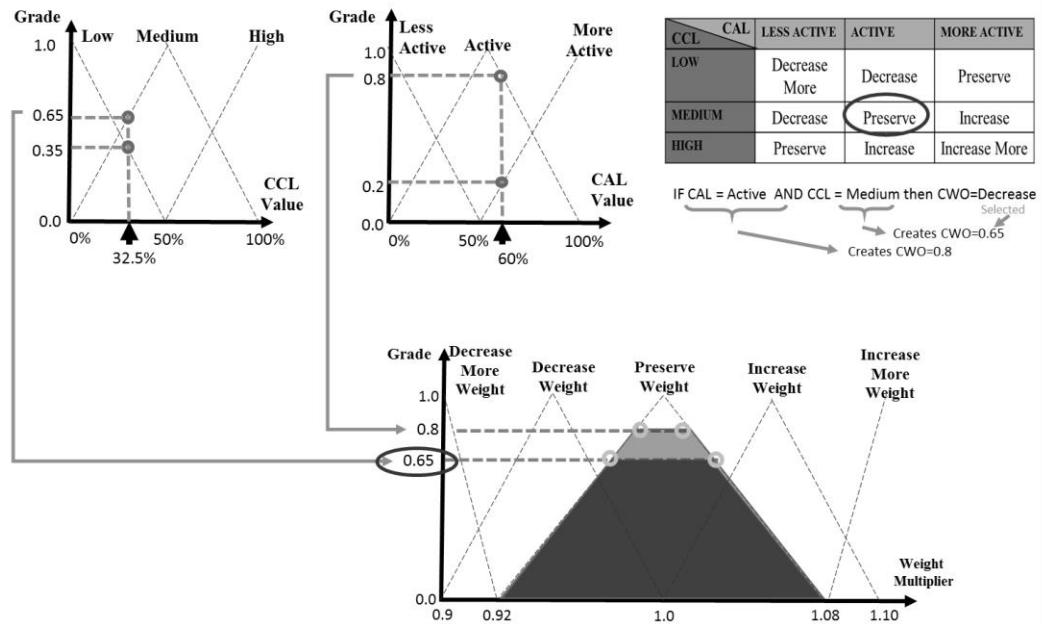


Figure 4-21 The third triggered rule with corresponding mappings

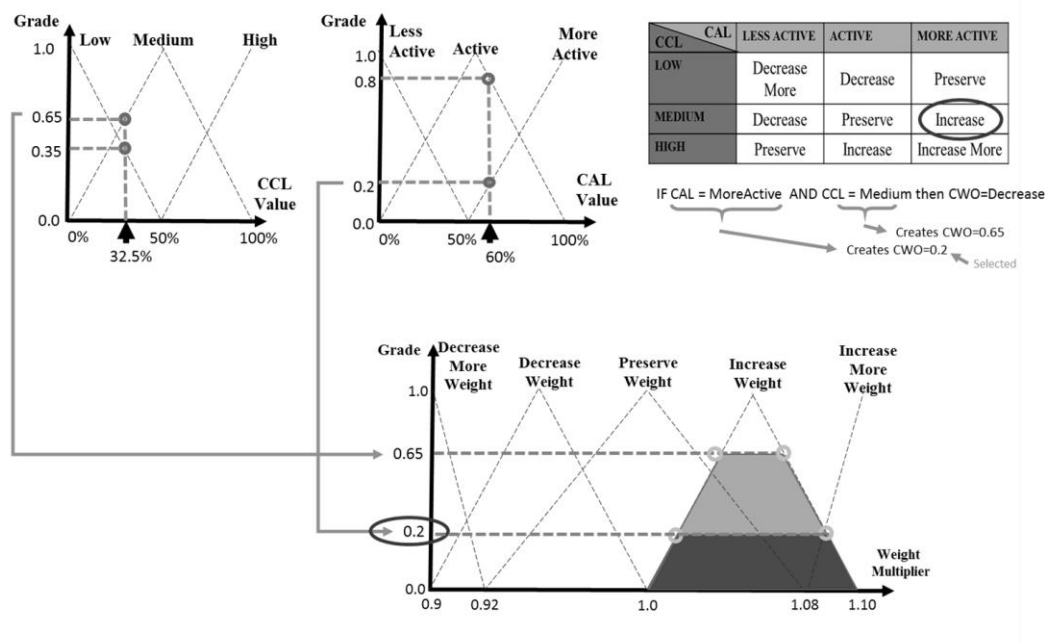


Figure 4-22 The fourth triggered rule with corresponding mappings

Finally, at defuzzification step, the all four outputs which are shown in Figure 4-23 are accumulated together and Figure 4-24 is obtained. Then centroid of these

accumulated outputs are calculated which is 0.995. This output value then converted into an action according to pre-defined actions listed in Figure 4-25.

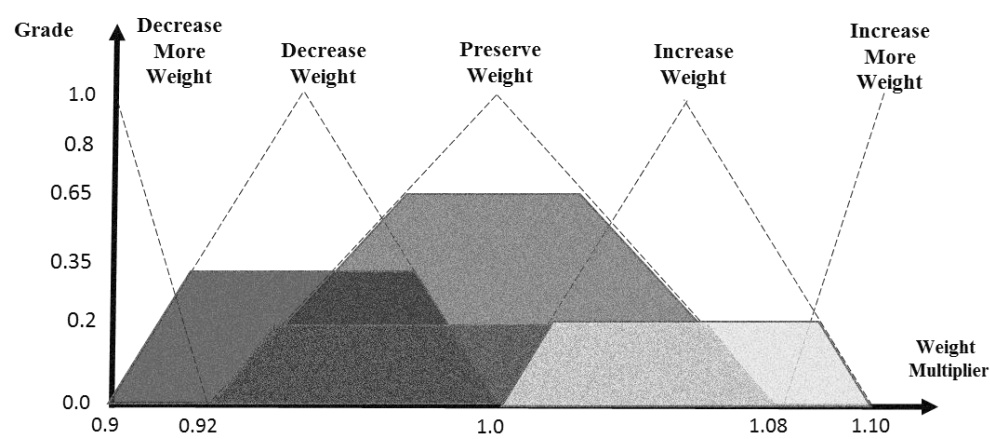


Figure 4-23 The outputs that are obtained from triggered rules

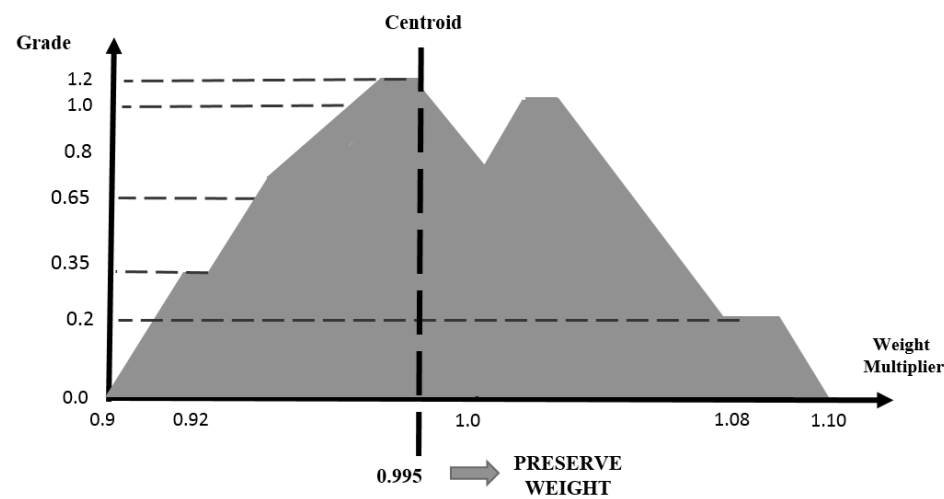


Figure 4-24 The accumulated output of fuzzy processing step and application of centroid method for defuzzification

```

if (outputValue < 0.940)
    DecreaseMoreWeight
else if (outputValue < 0.970)
    DecreaseWeight
else if (outputValue < 1.030)
    PreserveWeight
else if (outputValue < 1.060)
    IncreaseWeight
else
    IncreaseMoreWeight

```

Figure 4-25 The output of WB-FIE to action mapping

The steps that are just described are repeated for each active criterion in a periodic fashion which let system to make use of this feedbacks to improve overall prefetching performance.

4.6 Ensemble Adaptive Prefetching Approach and Usage Scenario

4.6.1 EAP Approach

After having described the architecture of EAP and its important concepts, now EAP approach and a general usage scenario are going to be described through an example application type using these concepts. The general usage is illustrated in Figure 4-26.

There are two important actors in EAP; EAP framework and application. The application is abstracted from EAP framework through provided unified interfaces. It can be considered as a component or library that can be integrated into any application. As illustrated in Figure 4-26, the requests are initiated by registered set of criteria and are loaded from provided data source. In this study, file system is used as data source through set of files, but obviously this can also be replaced with web servers. The set of criteria stay between application and EAP framework plays the interface role. Moreover, a common interface is provided to registered criteria and application to observe and change the currently employed prefetching parameters during execution time.

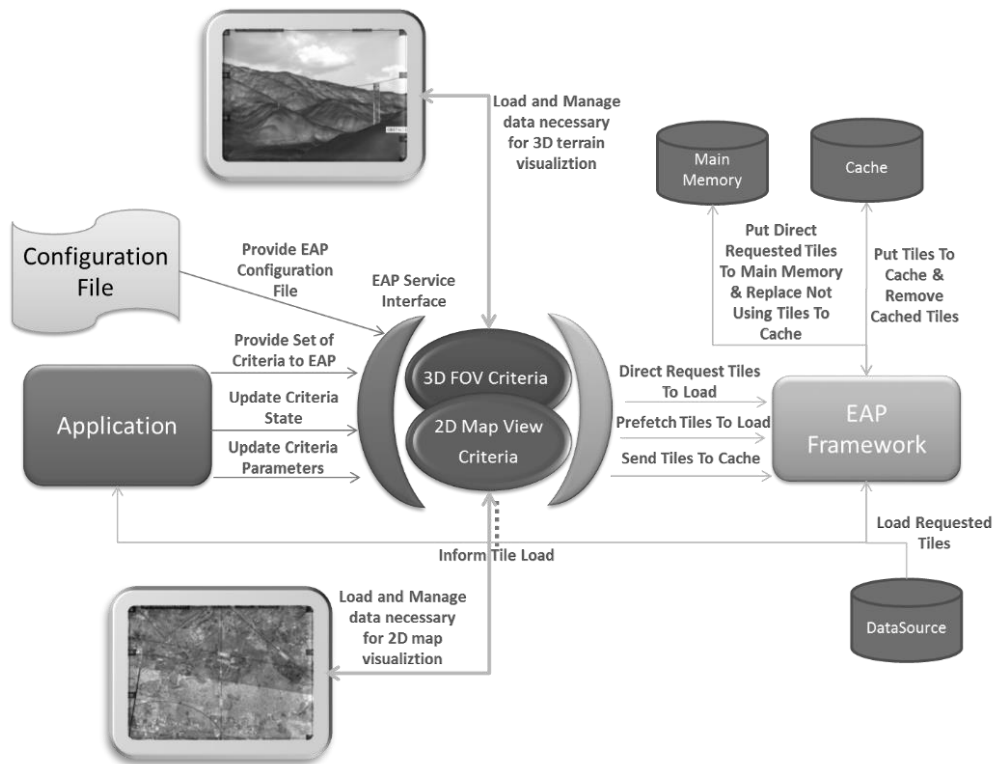


Figure 4-26 Overview of EAP usage

The main application responsibility is to register criterion to EAP, because the criterion is managed by EAP framework. Then it set necessary criterion prefetching parameters and provide application state data to these set of criteria through provided interface.

The other important application responsibility is to utilize user input and provide parameters to EAP framework. To help the applications and unify the inputs that are provided through criteria, navigator concept is introduced. For instance, for 2D Map View criterion, the main responsibility of navigator is to provide geographic location, speed, and altitude and zoom level. For 3D FOV criterion, it is expected to provide the field of view angle, view direction, rotation angle and etc. In addition to these parameters, navigator can also be used to provide these parameters through making use of keyboard and mouse.

This *configurability* is another important capability provided by EAP approach. The main motivation behind this configurability is to let application to be able to configure criterion, EAP framework parameters accordingly to current state of application and let also applications to interfere with EAP manually.

Moreover, user can either determine the list of active criteria before execution or she may activate/deactivate corresponding criterion during run-time. For instance, consider the case that is given in introduction part. The most of contemporary GIS applications begin to exhibit also 3D flythrough or terrain visualization capabilities as well as providing 2D map display and analysis capabilities. Although these capabilities have some common prefetching behavior, these three capabilities are considered as system modes. So whenever user switches to 3D mode, criteria that are developed for 3D might be activated and whenever user switch to 2D mode corresponding criteria might be activated.

Alternative to previously given system mode example, the application may also employ a set of criteria and change the weight of these criteria without deactivating any of them. With employed criteria and this weight infrastructure application can change the effectiveness of each criterion during runtime either manually through some predefined parameters like hit ratio, correctly and incorrectly predicted raster data tiles. However, it should be noted that the EAP performs this automatically through developed WB-FIE and criterion weighting, but such manual intervention is also permitted.

The other EAP parameters that can be configured by application at execution time are;

- The *direct* and *prefetching request* trigger threshold values,
- The number of loading threads,
- The cache and active tile container size,
- The prioritization policy parameters,
- The WB-FIE parameters (membership function parameters),

- The criterion weight,
- The criteria specific parameters.

4.6.2 EAP Usage Scenario

After having described the overall EAP approach, now a usage scenario is going to be described through a hybrid application with important steps. Some of these steps are related with the architecture given in section 4.1 through enumeration. The Digital Moving Map application is selected to be used for this purpose [41, 42]. The concept behind this application type is to represent the geographic area in which an aircraft's position is depicted and its environment is visualized and provided to pilot. As the aircraft's position changes, the area of map is also adjusting itself to keep up with the aircraft's progress along its flight path [43].

Digital moving map applications are developed to replace cumbersome paper maps in aircraft cockpits. They provide navigational and tactical information that is useful for performing corresponding tasks. In addition, they provide a means for enhancing mission effectiveness and situation awareness. They allow the pilot to focus her attention on navigation tasks with a minimum effort. These systems integrate information from several sources. They serve to display information more efficiently such that the pilot should be able to obtain all the information required to assess a situation and accomplish a task with a quick glance at the display. Moreover, digital moving map applications provide control of the displayed information to pilots [44, 45].

This type of applications require data for different kinds of domain specific operations at the same time. They are providing 2D map, 3D terrain visualization, analysis capabilities and most of these are running on avionics systems that have very limited resources. So it is very important to provide a prefetching mechanism that fulfill the data availability problem for these exhibited different kinds of capabilities and operations at the same time. Hence, it is also a good candidate for illustrating the

usage of EAP and how it can fit for these types of applications.

The usage scenario is described through phases which also summarize life cycle of digital moving map application.

4.6.2.1 Pre-execution Phase

In this phase, the activities that are required to be performed before application execution are going to be described. There are two important activities at this phase which are preprocessing and initial configuration file preparation (0 and 12 in Figure 4-1). The preprocessing activities are described in section 4.4.1 and these are usually common for all types of applications, so those will not be repeated here. The other activity is the preparation of a configuration file according to application requirements. This configuration file should contains settings about the data, data loaders, criteria and statistics. The format of this configuration file and the file prepared for this usage scenario is given in Appendix B.

The data that is going to be used by application is highly dependent on the application requirements. The digital moving map application requires at least elevation raster data and satellite photo data. So, configuration file should contain descriptive information about this data such as its resolution, path, format, and extent and tiling information. The EAP can handle multiple layers and data sources. As a result of having different types of data, the data loaders that are going to be used should also be provided through this configuration file. The EAP framework provide a generic interfaces and base classes to handle these different kinds of data, so user can develop its own data loader objects using these classes and then integrate them into EAP framework.

Another important configuration set is about registered criteria. For digital moving map, 3D FOV, 2D Map View and Vertical Cross Section (VCS) Analysis criteria are used for prefetching and related operations. There are some settings common to all

criteria such as the data and data loaders that will be used for these criteria, operations that are exhibited by each criterion, initial criterion weights and criterion operation thresholds. The user can enable or disable prefetching, requesting, replacement or evaluation operation from these settings. In addition to these settings, users can provide criterion specific parameters. For instance, for 3D FOV criterion, initial bearing, roll, pitch, view direction, view angle and depth parameters should be provided. For 2D Map View criterion, the map extent size and initial scale values should be provided and for VCS analysis the initial analysis length parameter should be provided. These settings are provided through pre-defined templates. Whenever a new criterion is introduced and registered, the corresponding set of parameters should be added into configuration file through given format. These configuration settings are then passed to each criterion at the beginning of application.

There is also a set of settings for statistics. The EAP provides a statistics infrastructure to applications to let them collect various parameters through application execution. These parameters are registered through this configuration file. At execution time these metrics are being collected and dumped into files to be analyzed after execution is completed.

4.6.2.2 Execution Phase

After configuration file is prepared and application is started, configuration settings are passed to EAP framework and initialization is performed with these settings. Then the execution phase is started where most of the activities described previously are occurred. Before examining the activities, capabilities of digital moving map application are going to be described. These capabilities resemble the criteria mentioned at the beginning of the section which are 2D map display, 3D terrain visualization and elevation analyzes.

The primary purpose of 2D map display is to provide pilots necessary information about its location, flight trajectory and of course situational awareness. Prior to digital

map systems, paper maps are being used extensively for this purpose. However, they are hard to follow while controlling the aircraft and very cumbersome. The 2D map displays renders raster tiles to pilot through display devices. Moreover, there are some additional operational and tactical layers for pilot usage. These tiles are brought together to obtain 2D map display. Example 2D map displays from different digital moving map applications are shown in Figure 4-27 and Figure 4-28.

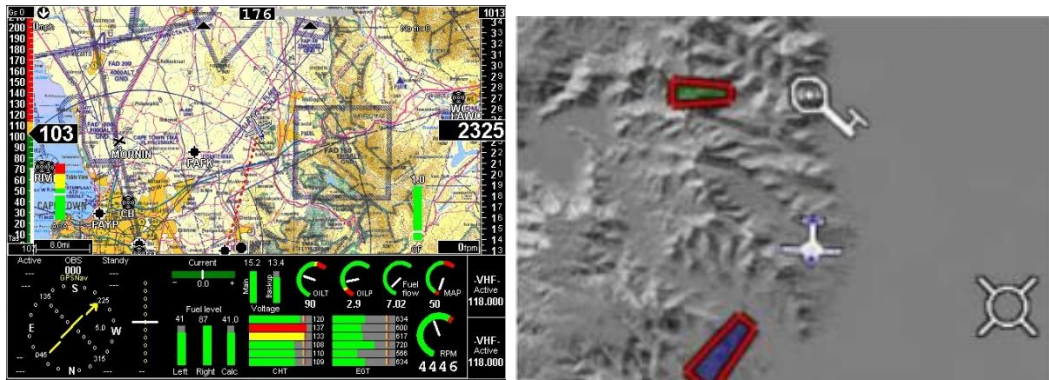


Figure 4-27 Example digital moving map application 2D map displays [46, 47]



Figure 4-28 Example digital moving map application 2D map displays (cont.) [48]

In addition to 2D map display, now these type of applications are also employing 3D terrain visualization. This 3D mode is being used especially for the cases where pilot is unable to determine its location or position because of weather conditions. It enhances terrain awareness. It is also being used for landing and takeoffs. Moreover, other elements such as airports, obstacles are being displayed and placed over 3D terrain. To be able to visualize the terrain, application take the elevations tiles and

generate terrain geometry using these elevation tiles. Then this terrain is being covered with satellite photo tiles to make it resemble the real terrain. There are various visualization techniques which makes use of these tiles. Although these techniques may use different methods, all of them requires elevation data and uses a visibility based occlusion. Example 3D terrain visualization displays from different digital moving map applications are shown in Figure 4-29.



Figure 4-29 Example digital moving map application 3D map displays [46, 48]

Finally, analysis capabilities are employed in these type of applications. The most important analysis is VCS analysis which is going to be addressed in this study. This analysis is being used to illustrate the terrain heights dynamically in the direction of aircraft so that pilot can assess the dangerous elevation heights and maneuver accordingly. An example display of VCS analysis is shown in Figure 4-30.

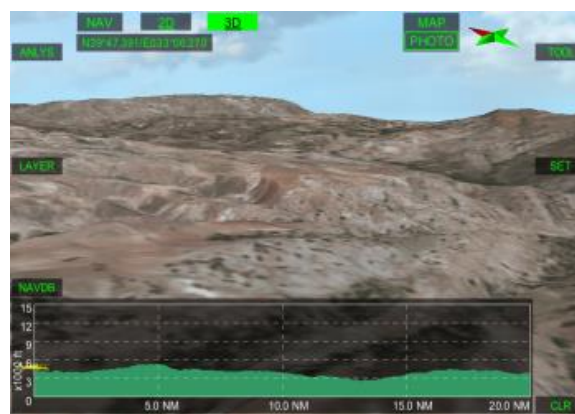


Figure 4-30 Example digital moving map application VCS analysis display [48]

For 2D, 3D and analysis capabilities, the prioritization policies described in section 4.4.3 are being used. At the beginning, usually no background information exists for prefetching, so direct load requests are initiated to load necessary tiles into memory (The sequence: 9-7-8-5-1-3 in Figure 4-1). As aircraft moves or user starts to interact with 2D map, the aircraft parameters like its location, heading and speed are provided to EAP framework through these three registered criteria. The corresponding criteria use these parameters to perform evaluation, prefetching and requesting operations.

In digital moving map application case, assume that it is started in 2D map mode. So necessary direct load requests are initiated to load those tiles that lies in client window to display 2D map to pilot. Whenever aircraft started to move or user started to pan or zoom, the registered criteria also started to initiate prefetching tile requests. For this purpose, 2D map view criteria initiate prefetching operation according to previous user action (The sequence: 9-7-8-5-1-2 in Figure 4-1). For instance, if user pans the map in the right direction, this criterion initiate prefetching load requests to load those tiles that lies in the right outside of map extent as illustrated in Figure 4-31. So whenever user pans to right, the corresponding tiles are provided to criterion immediately action (The sequence: 9-7-5-2-3 in Figure 4-1) and user do not realize any loading or delay which is the purpose of prefetching. In 2D mode, all other active criteria requesting operations are being disabled.

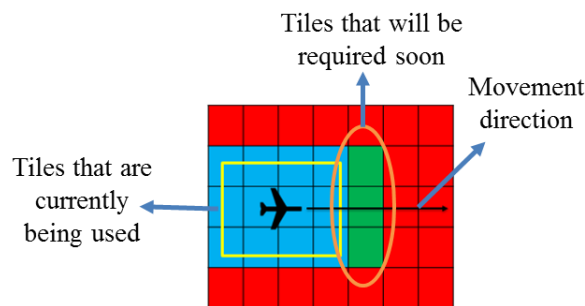


Figure 4-31 Tiles that will be prefetched by 2D map view criterion

In addition to 2D map view criterion, the other criteria also initiate prefetching load

requests as described in section 4.4.2 and 4.4.3. Moreover, other registered criteria evaluate the tile requests that are initiated by each other to come up with overall load order. For instance, prefetching initiated tiles that are also lying in the aircraft FOV will get higher priority values and being loaded earlier than other tiles. As user continue to use 2D map display and exhibit corresponding capabilities, its weight is being increased by WB-FIE and other criterion initiated prefetch tile requests are moved toward the end of load list. Here, it is important to note that WB-FIE is doing these through using the CCL and CAL values (8 in Figure 4-1). Hence, if user does not perform any action or perform too much unexpected movements, then there may not be too much increase in criterion weight. This is natural result of such adaptive behavior, but effects of unexpected navigation behavior is alleviated by employing a criterion that is designed for this purpose.

As size of cache reach to a predefined limit, replacement operation is being triggered to provide space for new tiles. Here, the cached tiles are being prioritized again by active three criteria and given number of tiles are being disposed. The limit of cache is provided through initial configuration file. The order of these cached items are usually highly dependent on the current mode of application. For instance, if user switch to 3D mode from 2D mode then the tiles that are being put to cache by 3D FOV criterion are less likely to be disposed.

Different from other two criteria, analysis criterion does not perform any operation till it is being enabled by application and when it is enabled then it also starts to initiate prefetch, requesting, evaluation and replacement operations. This is also a good example of how a criterion can be utilized through provided settings. In other words, applications can enable or disable specific criteria according to application requirements dynamically at runtime without manual intervention.

Whenever pilot switch to 3D mode, then requesting operation of 2D map view criterion is being disabled as 2D map display is not required anymore. The direct load request are started to be initiated to load the tiles that are necessary for terrain

visualization which are usually the ones that are inside the aircraft FOV. The capabilities that are exhibited in 3D mode are being performed according to aircraft navigation or behavior. For instance, whenever aircraft rotates, the FOV direction is also being changed or its location or its altitude. During 3D mode, 2D map view criterion still initiates prefetching requests and usually the ones that are also lies inside the FOV being loaded prior then others.

CHAPTER 5

DEVELOPED SET OF CRITERIA

Another important contribution of this study is to develop different criteria to increase the prefetching performance for given application domains in conjunction with corresponding capabilities as described in the introduction section. Moreover, this illustrates the possible use of prefetching techniques through EAP. In this section, developed criteria are described by their characteristics and usage in examples, according to the formalization given above.

5.1 2D MapView Criterion

This is the basis criterion for most of the 2D GIS and visualization applications. This criterion is developed to handle 2D map displays and corresponding user navigations. It has a deterministic nature and performs *Requesting*, *Evaluation*, *Prefetch* and *Replacement* operations. The main responsibility of this criterion is to initiate load request for raster tiles that lie in map extent or client window for 2D map rendering.

It is defined by a map extent, which has a center point, width, height, and rotation parameters in geographic coordinates and makes use of the 2D navigation prioritization policy mentioned in the query step. The tiles that are being loaded through direct load requests are determined according to these parameters. These parameters are illustrated in Figure 5-1. It prioritizes the raster tiles according to their distance to map extent center and whether they are inside the map extent or not. So if a raster tile is inside this map extent and located at the center, then it has the highest priority. This priority decreases as tiles getting further away from center. This criterion also evaluates the candidate raster tiles that were requested by other criteria

according to aforementioned distance based prioritization. In 2D navigation case, this criterion checks requests initiated by other criteria and may also discard some of these to prevent any unnecessary tile requests.

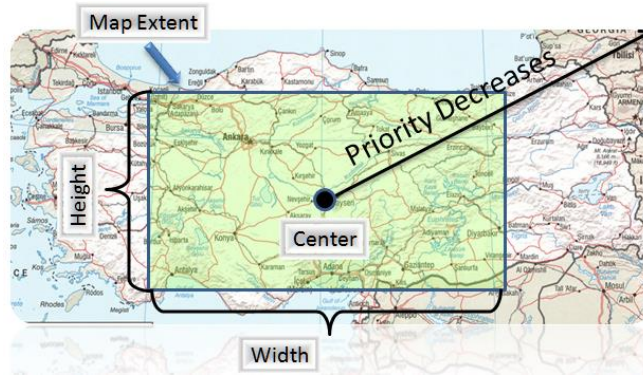


Figure 5-1 Overview of 2D Map View criterion

The trigger for requesting and prefetching operations is the change of center for given threshold (τ_R and τ_P) value. This value is initially given in configuration file and can also be changed during execution time. So whenever user pans or moves the map extent with given amounts, the actions are executed. It is either driven by aircraft's location or navigated through controllers like an observer in visualization applications or mouse panning in GIS applications. *Requesting* operation ensures that raster tiles that are within the map extent are ready whenever it is requested by application; however, if they are not, then it initiates direct load requests for them. When these tiles are being loaded, they are directly being put into main memory for application usage.

The *prefetching* operation is again performed according to aircraft/user movement (or panning) in such a way that if aircraft/user moves towards east then it prefetch the tiles enumerated as X, Y, Z and if it moves toward south west then A, B, C, D, E, F and G tiles are prefetched as illustrated in Figure 5-2. It is important to state that different from some studies like [4], the order of tiles that are going to be loaded are determined at runtime dynamically according position of aircraft/navigator. In [4], if user moves towards east the Y, X, Z tiles are loaded always with given order as shown

in Figure 5-2-a. However, in EAP approach, according to position of aircraft/navigator and other employed criteria this order may change. For instance, if navigator located near tile 2, the order will be X, Y and Z as shown in Figure 5-2-b.

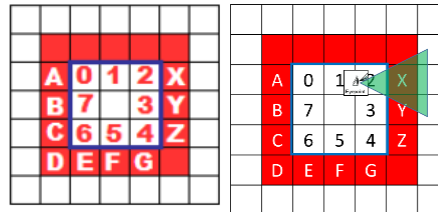


Figure 5-2 a) The fixed order of tiles that will be prefetched in east and south west direction [4], b) the dynamic order in EAP that determined according to navigator location.

5.2 Retrospective Adaptive Prefetching (RAP) Criterion

This criterion implements the RAP method [4] which is described in related work section in detail. It contains both 2D map navigation and zooming capabilities in itself. It performs *requesting*, *evaluation*, and *prefetch* and *replacement* operations and has a heuristic based nature.

This criterion is added to be able to use corresponding prefetching technique in EAP and also illustrate the possible usage of existent prefetching techniques in EAP through criterion mechanism. The requesting and prefetching behavior is same as the 2D map view approach which is to display 2D map.

5.3 2D/3D Distance Criterion

This criterion performs only *evaluation* operation and it does not initiate any direct or prefetching load requests. This criterion can be employed for different kinds of applications from GIS to visualization applications. It evaluates the raster tile requests initiated by other criterion and prioritize them according to their 2D distance to the

chosen pivot point. For instance, 2D map view criterion performs requesting operation and initiate load request for some candidate raster tiles. This criterion may evaluate their distances and assign *extra* priorities according to their distances to the pivot point. However, different from 2D map view criterion this distance is not related with the client's window or map extent. With this criterion, applications assign different pivot points and makes use of them in prioritization in addition to the navigation position. This criterion can also adapted for 3D space with the aid of altitude parameter. An overview of this criterion with pivot point and map extent center is depicted in Figure 5-3.

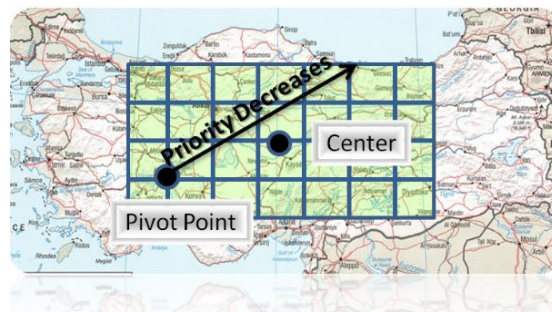


Figure 5-3 Overview of 2D Distance criterion and pivot point

5.4 3D Field of View (FOV) Criterion

The 3D FOV criterion is developed for 3D applications that require visibility-based prefetching and visualization capabilities. This is especially critical for applications that employ FOV-based LOD or data models like terrain visualization. It is defined through FOV angle, FOV depth, bearing, and pitch parameters, and it performs *requesting, evaluation, and prefetch and replacement* operations.

The main responsibility of this criterion is make sure that every raster tiles that lies in the range of 3D field of view ready and loaded to be used by application. It is deterministic and does not employ any heuristics. It is defined through field of view parameter, FOV depth parameter, bearing and pitch parameters as illustrated in Figure 5-4. This criterion makes use of the 3D FOV prioritization policy mentioned

in the query step.

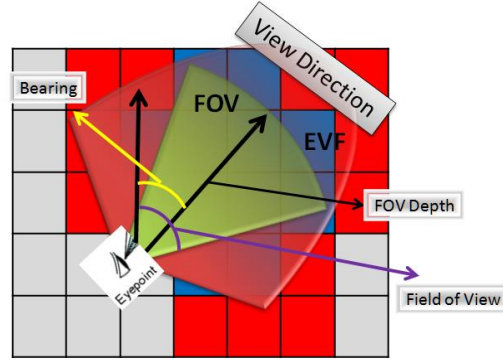


Figure 5-4 Overview of FOV Criterion and its parameters

The trigger for *requesting* and *prefetching* operations is the change of aircraft position or rotation of view direction (i.e. bearing or pitch) for given threshold (τ_R and τ_P) values. Similar to other criteria, these values are provided through initial configuration file. So whenever navigator rotate its FOV or move for given amount the corresponding requesting and prefetching operations are performed.

For FOV prefetching operation, *EVF* (*Extent View Frustum*) concept is introduced. The EVF represent the area that will be probably required to be loaded very soon. It is obtained through FOV parameters, navigator speed and view frustum rotation direction. The difference between the FOV and EVF region represent the area that will be used for prefetching raster tiles. Also the rotational speed of viewer is employed for this purpose in such a way that a faster rotational speed causes a larger EVF generation.

For instance, in Figure 5-4, the blue raster tiles represent currently active tiles and red ones represent the tiles that will be prefetched. It also evaluates already initiated raster tile requests according to mentioned prioritization policy.

5.5 Point of Interest (POI) Criterion

Most of previous prefetching techniques interest in requesting raster data tiles to

render a 2D map or 3D terrain. However, for some applications, especially command and control or planning applications, the data which is located around a given point of interest might be more important and needed to be loaded before the other raster tiles. This criterion is developed especially to fulfill these requirements. It performs *requesting*, *evaluation*, and *prefetch* and *replacement* operations.

The main responsibility of this criterion is to initiate *direct load requests* that cover the point of interest and the area around that point. It also evaluates the current raster tile requests whether they lie inside this region or not. The prefetching operation is performed in such a way that whenever aircraft/navigator comes near to given POI and satisfies the determined spatial distance threshold, it initiates *prefetching load requests* and load those tiles into cache according to corresponding. The range of POI is specific to each data and criterion performs its operations accordingly. Finally, whenever aircraft/navigator goes out the scope of POI, the loaded raster tiles are being sent to cache.

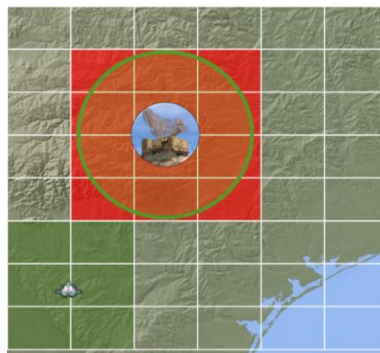


Figure 5-5 Point of Interest criterion usage for radar POI

In Figure 5-5, the green raster tiles represent currently active tiles and red ones which centered at given radar POI represent the tiles that will be prefetched.

5.6 Vertical Cross Section (VCS) Analysis Criterion

Last developed criterion is vertical cross section (VCS) analysis, which is developed

primarily to fulfill VCS analysis requirements. Although one of previously mentioned criteria may be employed for this purpose, these may cause unnecessary raster tile loads. Because, in VCS, only the height values that are in the current view direction are required; thus, other data does not required to be loaded. It performs *requesting*, *evaluation*, and *prefetch* and *replacement* operations.

The requesting operation loads the raster data tiles that are across the vertical cross section, and the prefetching operation prefetches the tiles that are across the extension of analysis track. The length of analysis is dynamic and can be configured during execution time. This criterion makes use of the analysis prioritization policy mentioned in the query step. The overview of VCS analysis criterion is depicted in Figure 5-6.

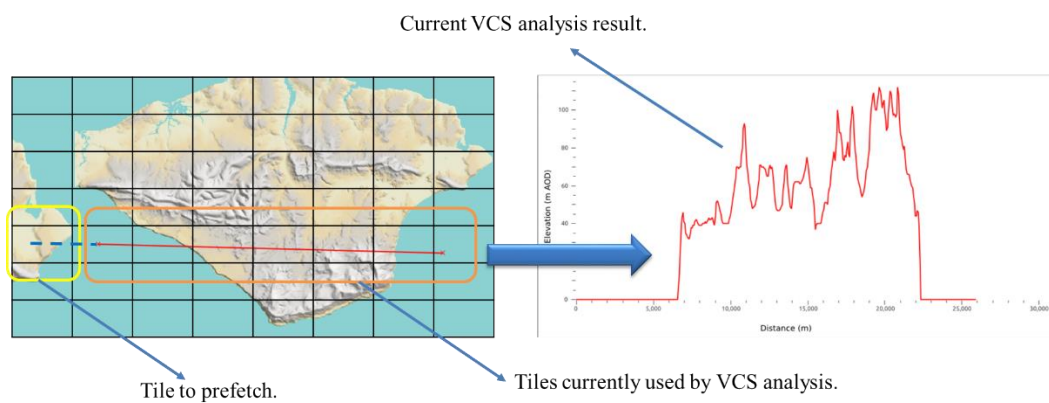


Figure 5-6 Overview of VCS analysis criterion

CHAPTER 6

EXPERIMENTATION AND DISCUSSION

In this section, measurement parameters and how EAP performs under these parameters are described. First of all, measurement parameters are described. Then an experiment which is based on a flight trajectory is designed to be used with EAP to illustrate its performance in detail with different scenario executions. In addition to this flight trajectory, additional experiments are also being conducted to show performance gains obtained in different scenarios. The results obtained from these experimentation are also given in corresponding section. Finally, chapter is concluded with discussion about measurement results and overall EAP approach.

6.1 Measurement Parameters

The first parameter is hit ratio (HR). It is defined as correct predictions to overall number of requests, which is given in equation (4) [49]:

$$\mathbf{HR} = (\text{No. of Correct Predictions [hits]}/\text{Total No. of Requests}) * 100 \quad (4)$$

HR itself is a good representative of overall prefetching performance. However, to assess individual criterion performance an additional parameter, i.e., correct prediction ratio (CPR), is employed. CPR represents the ratio of correctly prefetched tiles to total number of prefetched tiles of corresponding criterion, which is calculated by equation (5):

$$\mathbf{CPR} = (\text{No. of Correct Predictions [hits]} / \text{Total No. of Predictions}) * 100 \quad (5)$$

Two more parameters are defined for WB-FIE, which are CAL and CCL. The CPR parameter is used directly for CCL. The CAL parameter is obtained by observing the

activities of individual criteria. For instance, in a 3D FOV case, if the FOV bearing or angle changes frequently, then the 3D FOV criterion activity level will increase. Similarly, in a 2D case, if the user pans continuously, the activity level will increase. The activity level decays by time if no activity occurs. If any activity occurs, then the activity level is set to 50%. These activities and their contribution to CAL are determined according to the nature of each criterion. For a 3D FOV criterion, the FOV bearing, angle, depth, and location changes are considered as activities. The highest contributor is FOV bearing change with 80% with the remaining 20% shared among other activities.

6.2 Experimentation One

A comprehensive flight trajectory is designed for first experimentation. In this section, details about this experimentation, setup and obtained results are given under successive sections.

6.2.1 Experiment Setup

The given trajectory is specifically designed to show how ensemble usage of prefetching techniques increase the performance of overall hit ratio. This trajectory contains different navigation patterns, but in a more complete fashion. Design of this trajectory is done according to possible navigations that can be exhibited by aircrafts that employ hybrid applications like digital moving map applications. Although trajectory may appear fixed, it is prepared so that it does not only include 2D navigation habits but also 3D navigations and analyses. So the trajectory also contains various navigation habits.

During this flight, different navigation patterns (for 2D: panning left, right, zooming in/out; for 3D: turning right/left, ascending, descending, hovering, FOV/bearing/pitch changes) that are used in real world 2D/3D hybrid applications are induced randomly to measure the performance of EAP. Through this route, different modes, such as

2D/3D, and capabilities, such as analysis, are enabled. At the beginning of the route, 2D mode is enabled and corresponding panning activities are performed, then 3D mode is activated. Then, all activities and capabilities are enabled and exhibited until the end of the route. In addition to the aforementioned patterns, certain abrupt patterns are also introduced into route to observe the behavior of EAP. For instance, panning right then down, panning up and left, or changing the FOV bearing in the opposite direction rather than turning it smoothly.

The speed of the navigator is increased by a factor of 100 for simulation purposes; thus, it takes approximately 6 min to complete the designed trajectory. The total length of the trajectory is about 1500 km. The digital terrain elevation data (DTED) Level 2 is used as raster data, which has a resolution of 30 m [50], and is extensively used by both 2D and 3D applications. This raster data is preprocessed into 512 x 512 raster tiles. A 64 raster tile-sized cache, which is approximately 32 MB, is employed for these executions. The cache size is deliberately chosen so small to show the true performance of EAP in such restricted environments, because prefetching operations usually perform better with bigger caches. An overview of the designed flight trajectory is shown in Figure 6-1. The behaviors exhibited during this trajectory is illustrated on this figure.

First of all, the current system modes and capabilities that are activated through execution is illustrated with different line styles and described in top left box of Figure 6-1. From start till the 65 s, mostly the 2D map display related navigation patterns are exhibited like panning. Then till 100 s, only 3D visualization capabilities are exhibited and finally till the end of the course both of 2D/3D/Analysis system capabilities are exhibited altogether. During this trajectory, especially among 197 s and 270 s, the capabilities belongs to these three criteria are frequently exhibited to also test EAP under such stressed situations.

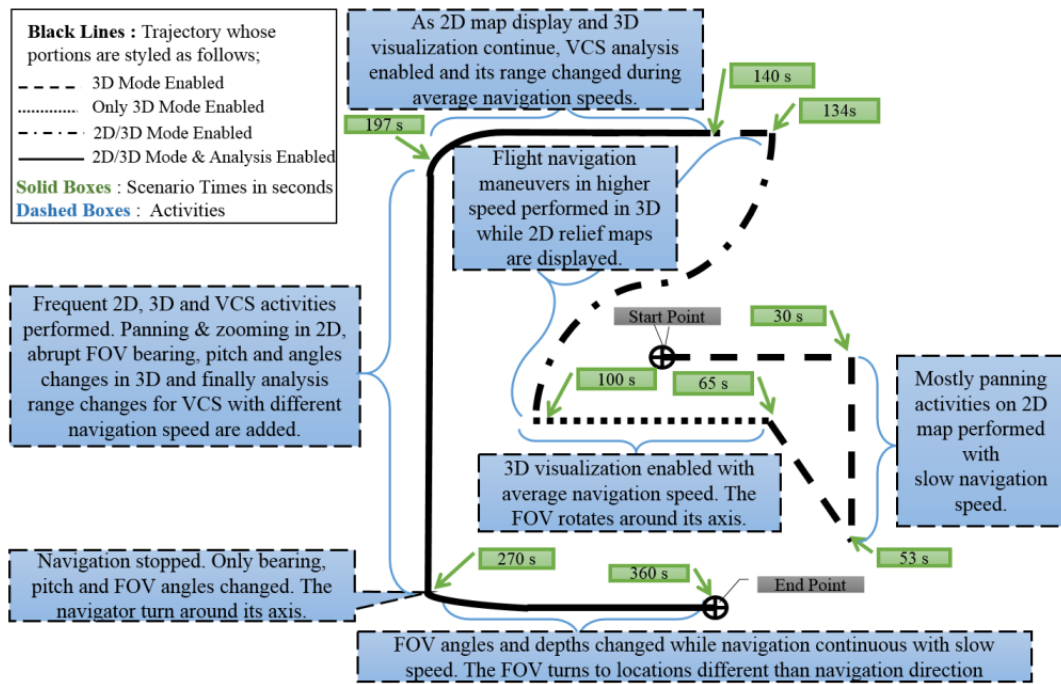


Figure 6-1 The designed flight trajectory that scenarios are executed through with corresponding behaviors exhibited through this path

6.2.2 Results and Discussion

Four scenarios are designed for experimentation. In the first two scenarios, individual 3D FOV and 2D map view criteria are employed to measure their individual prefetching performances, and to be used as a basis for comparison for ensemble usage. In the third scenario, the ensemble of 3D, 2D, and analysis criteria with fixed weights are employed to see how non-adaptive EAP performs. Finally, three criteria with adaptive mode enabled are employed to demonstrate the true performance of EAP. It is important to note that the route, activities, and capabilities in each scenario are exactly the same, except for the employed criteria and whether or not ensemble/single or adaptive mode is enabled. Figure 6-2 shows the overall HR results of each scenario execution.

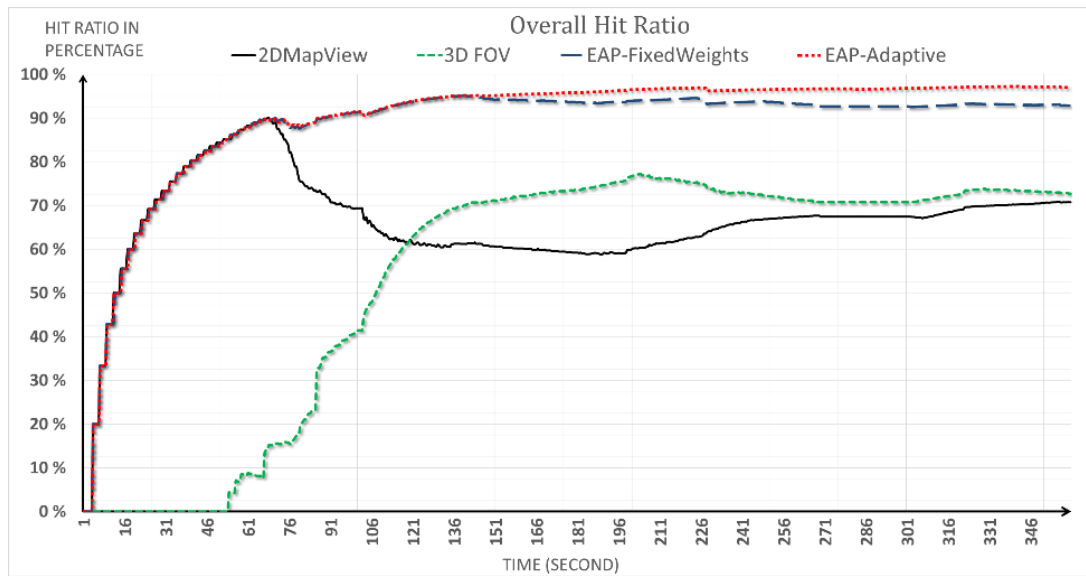


Figure 6-2 The overall hit ratios (HR) obtained from executed scenarios

It is illustrated that, although the newly developed 2D and 3D criteria achieve 71% HR individually, it increases to 92% with fixed-weight EAP, and finally to 97% with adaptive EAP. The adaptive EAP usage outperforms the individual cases by approximately 25%. Moreover, EAP preserves this high HR through the whole scenario execution, which is not the case for individual ones. For instance, although the 2D criterion achieves a high HR from the beginning to 60 s, it decreases as 3D capabilities are enabled, whereas the 3D criterion HR starts to increase. It can also be observed that such ups and downs in HR are well absorbed by EAP.

The given scenarios are also executed with a 16 raster tile-sized cache, which is about 8 MB. An HR of approximately 65% is obtained for individual 2D and 3D criteria, 81% for fixed-weight EAP, and finally 83% for adaptive EAP, which still performs better than individual usage. It should also be noted that fixed-weight EAP execution overall HR do not increase after a point. In fact, this is the point where all kinds of capabilities and navigation are started to be exhibited by user. On the other hand, the adaptive EAP handles this and even increase the HR about 5%.

The following figures show the parameters that are used and affected by EAP

adaptive mode. The two input values of WB-FIE, i.e., CCL and CAL, are illustrated in Figure 6-3 and Figure 6-4. The resultant weight values obtained from WB-FIE are shown in Figure 6-5.

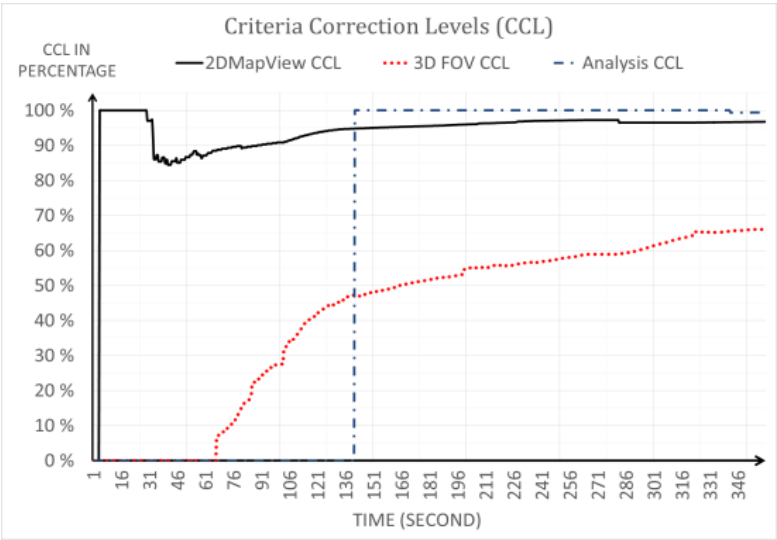


Figure 6-3 The CCL values that are obtained from adaptive EAP scenario execution

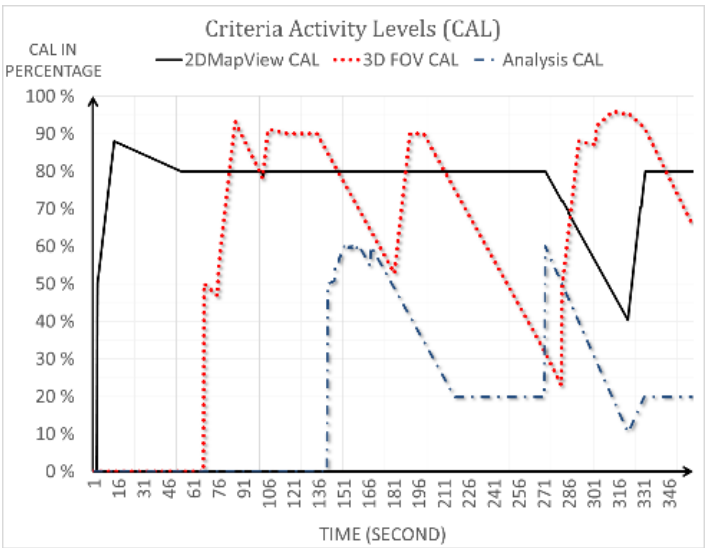


Figure 6-4 The CAL values that are obtained from adaptive EAP scenario execution

When CCL values are analyzed, the 2D criterion has higher CCL values than the 3D,

which indicates that 2D criterion has a better prefetching performance individually. On the other hand, the analysis criterion performs better than the other two criteria because it is very simple in its nature.

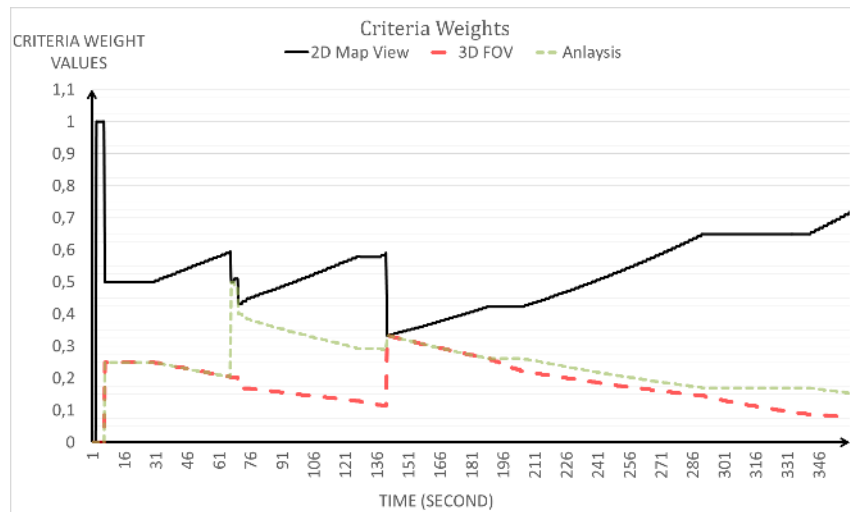


Figure 6-5 The criteria weight values that are controlled and changed by adaptive WB-FIS

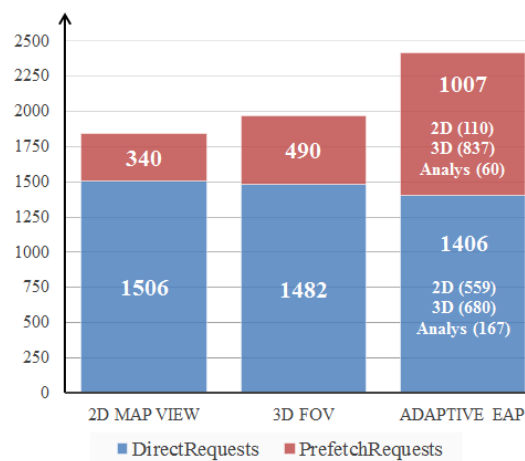


Figure 6-6 The number of direct (blue) and prefetch requests (red) initiated

When CAL values are analyzed, the 2D criterion maintains a higher activity value, which is primarily based on location changes. In fact, it increases as it moves and decreases whenever it is stopped, which can also be seen in Figure 6-4. The 3D criterion activity, on the other hand, is affected mainly by visibility-based parameter

changes rather than location changes. For instance, from 270 to 330 s, the location is fixed and the viewer's FOV angle, bearing, and its depth is continuously changed, which increases CAL values as it decreases 2D and analysis values. When these two values are analyzed, EAP favors 2D map view criterion because it has higher CCL and CAL values than the other two criteria, which can be seen with weight values in Figure 6-5. Therefore, although EAP reduces the weight of 3D FOV, a higher HR is achieved, which primarily stems from the fact that it started to make use of 2D criterion-predicted raster tiles rather than its own.

The 25% HR increase obtained could be improved further with additional criterion usage. One question that might arise at this point is the trade-off this approach. When programmatic calculations are analyzed, the processing overhead is negligible with respect to obtained overall hit ratio. The real overhead is the number of requests that have been initiated. Although three different criterion are employed in collaborative mode, the total number of requests does not increase, as illustrated in Figure 6-6. In fact, the total number of direct requests is decreased in EAP, at a cost of a slight increase in prefetch request counts. The biggest portion of prefetch requests come from the 3D criterion, which is also realized by the adaptive EAP system, and reflected as a reduction in weight of corresponding 3D criterion, as shown in Figure 6-6. Moreover, the prefetch request counts of the other two criteria also decrease. This increase in 3D criterion can be further decreased by more complex and well-performed individual criterion design.

One last point that should be mentioned here is the load order. The EAP not only provides a higher HR, but also reduces the user-perceived delays by providing the application with more required raster tiles as soon as possible.

Now, additional conducted experiments are going to be given in less detail to illustrate the performance of proposed approach in different scenarios.

6.3 Experimentation Two

In this experiment, another flight trajectory is being designed for experimentation. The details about this experimentation, setup and obtained results are given under successive sections.

6.3.1 Experiment Setup

The experimentation is again based on a well-known navigation pattern which is called Hold Pattern. This navigation pattern is being used by aircrafts to circle over given location. Design of this trajectory is performed by considering such navigations that can be exhibited by aircrafts that employ digital moving map applications.

Here, for given experiment, additional patterns are also injected into this trajectory as illustrated in Figure 6-7 which are 2D based like panning top-left, right, and bottom, top and bottom-left. At the beginning of the route, both 2D and 3D modes are enabled and when panning activities are being performed, 3D mode disabled and then enabled. Similarly, DTED2 is also being used as raster data in experimentation.

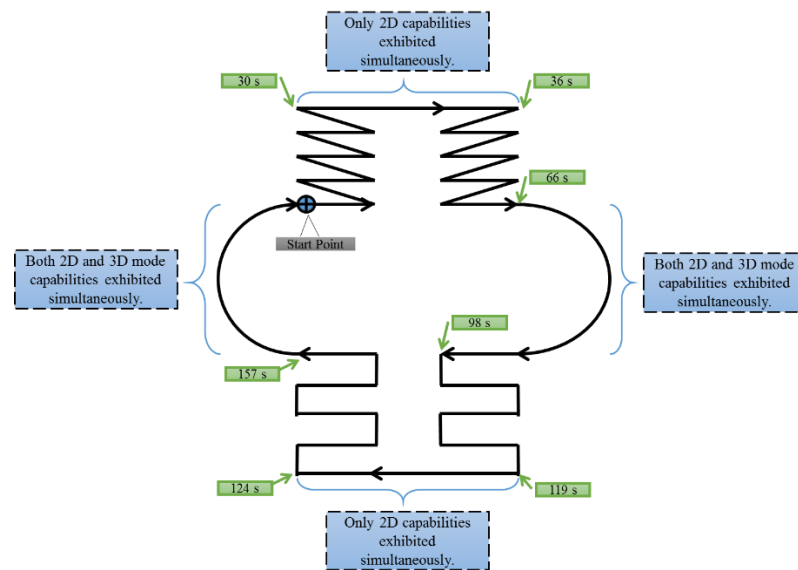


Figure 6-7 The designed flight trajectory for experiment 2

6.3.2 Results and Discussion

Three scenarios are designed for experimentation. In the first two scenarios, individual 3D FOV and 2D Map View criteria are employed to measure their individual prefetching performances, and to be used as a basis for comparison for ensemble usage. In the third scenario, the ensemble of 3D FOV and 2D Map View criteria with adaptive mode enabled are employed to demonstrate the performance of EAP. It is important to note that the route, activities, and capabilities in each scenario are exactly the same, except for the employed criteria and whether or not ensemble/single or adaptive mode is enabled. Figure 6-8 shows the overall HR results of each scenario execution.

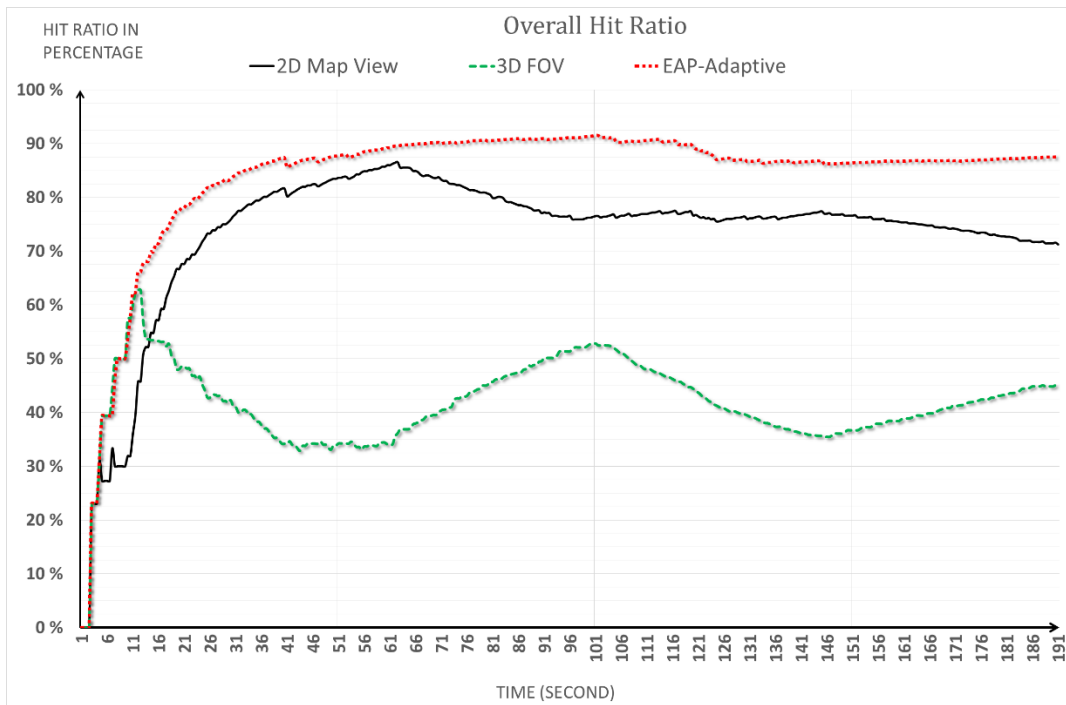


Figure 6-8 The overall HRs obtained from execution of experiment 2 scenarios

Although the newly developed 2D and 3D criteria achieve 72% and 45% HR individually, it increases to 88% with adaptive EAP usage. The adaptive EAP usage outperforms individual 2D Map View criterion by approximately 16% and 3D FOV over 40%. Moreover, EAP preserves this high HR through the whole scenario

execution as in first experimentation, which is not the case for individual ones especially for 3D FOV criterion. For instance, although the 2D criterion achieves a high HR from the beginning to 60 s, it decreases as 3D capabilities are enabled, whereas the 3D criterion HR starts to increase. The 3D criterion HR increases and decreases are too steep as a result of not exhibiting too much 3D capabilities comparing to 2D behaviors. In fact, the major difference between individual 2D Map View and 3D FOV is mainly stems from the behaviors exhibited through trajectory where most of the time 2D behaviors are exhibited.

6.4 Experimentation Three

Three different routes are designed for third experimentation. The details about this experimentation, setup and obtained results are given under successive sections.

6.4.1 Experiment Setup

The experimentation is performed through three different routes. These three geographic routes are designed deliberately to illustrate performance of EAP in different scenarios. It is important to note that the route, activities, and capabilities in each scenario are exactly the same, except for the employed criteria and whether or not ensemble/single or adaptive mode is enabled

As in other experiments, DTED2 is used as raster data for this experimentation. Different from other experimentations, another prefetching technique is implemented and employed for execution with these set of trajectories to both illustrate the possible usage of other prefetching techniques in EAP and also to illustrate the performance of individual criterion being developed. The RAP [4] technique is being used for this purpose. However, RAP criterion itself is not included in collaborative mode, because its purpose is same as 2D Map View criterion that was developed so adding it is not meaningful.

The characteristics and overview of first route is illustrated in Figure 6-9. In the first part of route (till the 45 s), the 2D mode is enabled and corresponding navigation capabilities are exhibited and then 3D mode is enabled and again corresponding capabilities are exhibited and FOV bearing angle is being changed continuously. This experiment designed to illustrate how different user navigation behaviors are handled by individual and ensemble usage in case of exclusive usage. In other words, these capabilities are not exhibited at the same time. The trajectory is approximately 122 km.

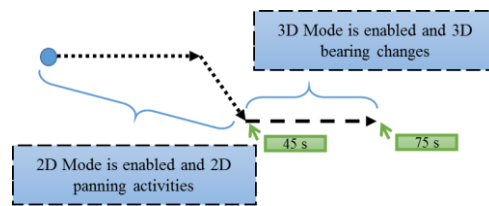


Figure 6-9 The designed first route for experiment 3

The characteristics and overview of second route is illustrated in Figure 6-10. In this route, both 2D and 3D mode capabilities are exhibited at the same time through all route. The abrupt directions changes are introduced to show how these case are handled. This route is longer than the previous one and is approximately 350 km.

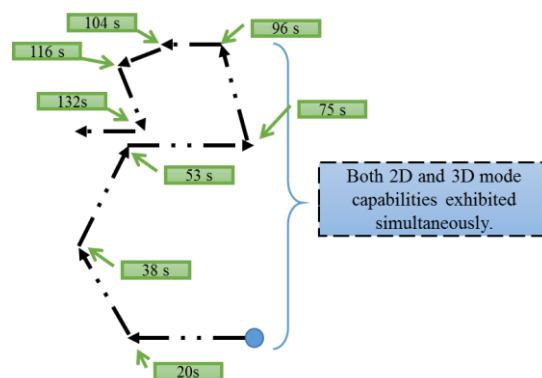


Figure 6-10 The designed second route for experiment 3

The characteristics and overview of third route is illustrated in Figure 6-11. In this route, in addition to 2D and 3D mode capabilities, analysis capabilities are also being

exhibited through this route simultaneously with other capabilities. The route is designed as zig-zag that contains forward and backward movements. This route is longer than the previous one and is approximately 210 km.

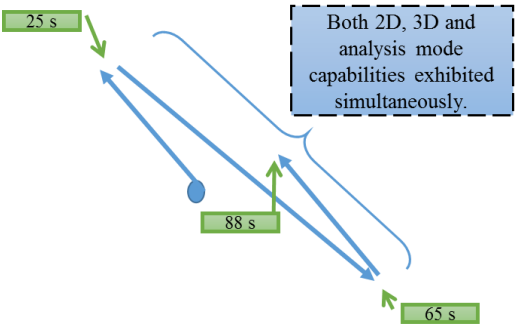


Figure 6-11 The designed third route for experiment 3

6.4.2 Results and Discussion

Four scenarios are designed for experimentation. In the first three scenarios, individual 3D FOV, 2D Map View and RAP criteria are employed to measure their individual prefetching performances, and to be used as a basis for comparison for ensemble usage. In final scenario, ensemble adaptive usage employed to show how EAP performs. In ensemble usage, both 2D Map View and 3D FOV criteria are being employed for route 1 and route 2. For route 3, additional analysis criterion is also being employed for EAP-adaptive usage.

Figure 6-12 shows the overall HR results of first route scenario executions.

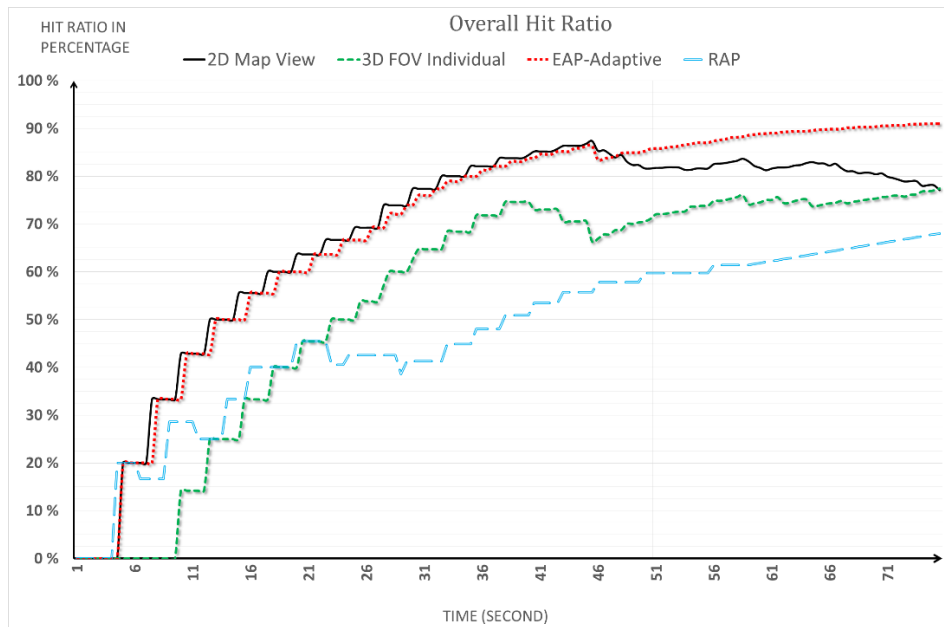


Figure 6-12 The first route executions and corresponding overall HRs

The individual results from route 1 executions shows that for capabilities exhibited in this route, the developed 2D Map View and 3D FOV criterion performs better than RAP criterion.

On the other hand, both of individual executions shows that till a point both criterion performs well but when 3D capabilities are started to be exhibited, 2D Map View criterion's HR started to decrease. Besides, 3D FOV criterion handles these capabilities and HR still continue to increase.

In EAP-adaptive execution, the ensemble usage performs approximately 15% percent better than individual prefetching technique usage.

Figure 6-13 shows the overall HR results of first route scenario executions.

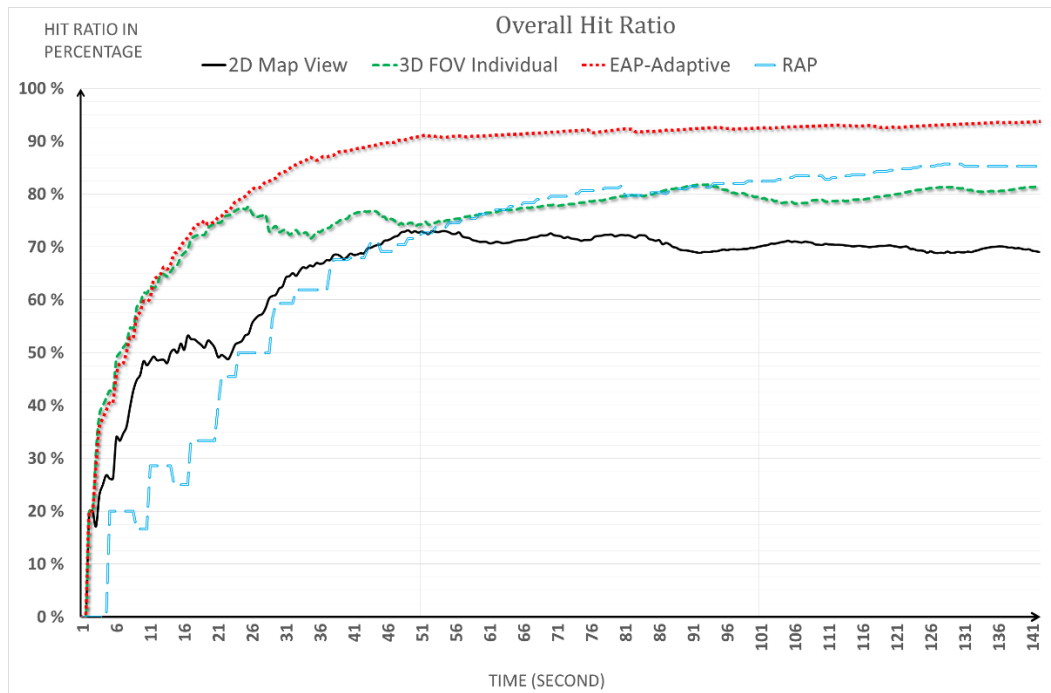


Figure 6-13 The second route executions and corresponding overall HRs

The individual results from route 2 executions shows that for capabilities exhibited in this route, the RAP criterion performs better than the developed 2D Map View and 3D FOV criterion. The main reason for this difference is believed to be the employment of heuristic method that make use of historical navigations.

Both of individual executions shows that after a point both individual criterion achieve a HR and then increases and decreases occur as execution continues. In other words, HR cannot be preserved or stable HR is obtained with these individual executions. On the other hand, EAP-adaptive execution both shows a much better HRs over individual executions and provides a more stable HR through execution.

Approximately 10% better HR over individual RAP criterion, 15% HR over individual 3D FOV criterion and over 20% HR is gained over individual 2D Map View with EAP-adaptive execution.

Figure 6-14 shows the overall HR results of first route scenario executions.

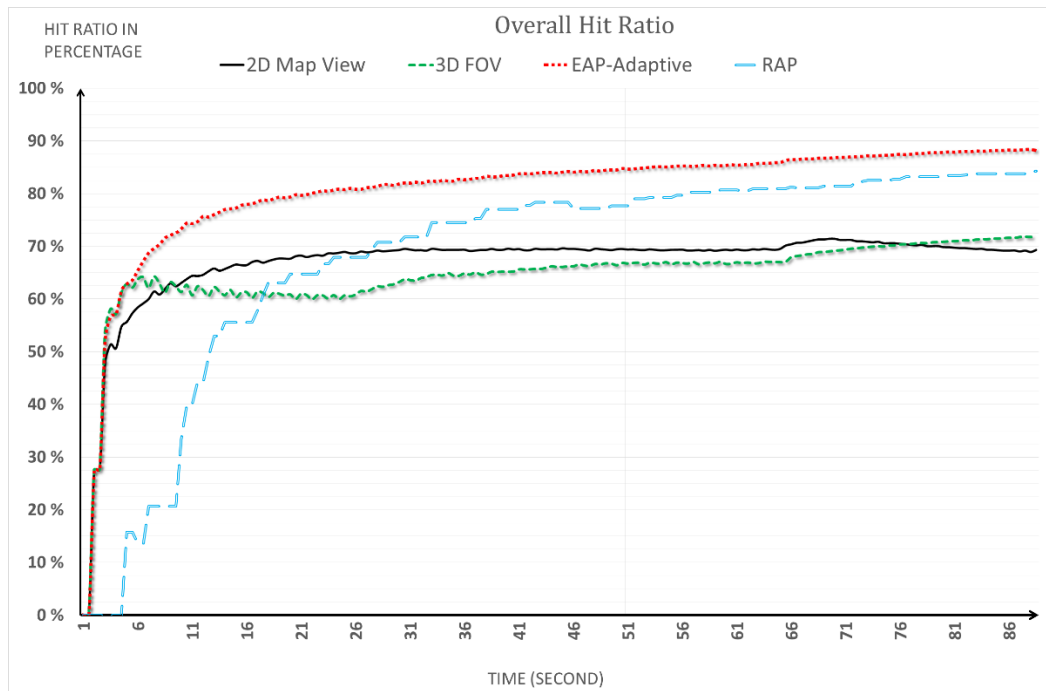


Figure 6-14 The third route executions and corresponding overall HRs

The individual results from route 3 executions shows that for capabilities exhibited in this route, the RAP criterion performs better than the developed 2D Map View and 3D FOV criterion as in route 2 case.

The individual executions shows a relatively stable HRs which become fixed around 70%. The main reason for this stable HR is the route itself as illustrated in Figure 6-11. On the other hand, EAP-adaptive execution still achieve a much better HRs over individual executions and provides a slightly more stable HR through execution. Approximately 20% better HR over individual 3D FOV and 2D Map View criteria and 5-10% better over individual RAP criterion is gained with EAP-adaptive execution.

All these three route executions shows that EAP performs still much better HRs than individual prefetching technique employment for different kinds of routes and user navigation patterns.

CHAPTER 7

CONCLUSION AND FUTURE STUDY

In this study, a priority based adaptive tile prefetching approach is proposed which enable to use ensemble of different prefetching techniques together for applications that exhibit different capabilities. Although there are various studies about prefetching, they are usually aiming to solve this problem by focusing on specific types of applications and these become insufficient as the boundary between these applications blurs. To overcome this problem, EAP is developed to let applications employ an ensemble of different types of prefetching techniques collaboratively in one unified model. The contribution of each prefetching technique is determined through a fuzzy logic-based weight balancer and prioritization.

To do so, first of all, the prefetching problem, its importance and current issues with contemporary approaches are explained. After outlining the related literature, EAP architecture, its important elements, and the prefetching process are described. Finally, developed prefetching techniques are described. It is demonstrated that EAP achieves better prefetching performance even with applications that exhibit different characteristics. Moreover, user-perceived delays are reduced with more precise load ordering. Besides, a cross-platform framework is developed to illustrate its real-world usage and lets other users to utilize given approach easily.

As a result of being generic in its nature, the EAP has many potential subtopics. One future study is to improve EAP approach so that it handles prefetching for geographic vector data. Although characteristics of vector data and raster data are very different, it is believed that EAP approach can also be used for this purpose.

Another future study is to adapt this method so that it can also be used at server side. However, there are some additional concerns in case of server usage like multiple clients and having less information about the nature of client applications. Moreover, previous client requests or other historical data can be employed for this purpose.

Finally, it is believed that the mechanism and infrastructure developed here could easily be used for 3D visualization and mobile GIS application especially out-of-core terrain visualization applications. More specialized prefetching techniques can be developed to fulfill the specific visualization requirements.

REFERENCES

- [1] Liqiang Z., Liang Z., Yingchao R., and Zhifeng G., “*Transmission and Visualization of Large Geographical Maps*”, ISPRS Journal of Photogrammetry and Remote Sensing, 66 (1), 73-80, 2000.
- [2] Hawick K. A., Coddington P. D., and James H. A., “*Distributed Frameworks and Parallel Algorithms for Processing Large-Scale Geographic Data*”, Parallel Computing 10, 2003.
- [3] Smith T., Menon S., Star J., and Estes J., “*Requirements and Principles for the Implementation and Construction of Large-Scale Geographic Information System*”, pp. 19-37 in W. Ripple (ed.) *Fundamentals of Geographics Information Systems: A compendium*. Bethesda, Md.: American Congress on Surveying and Mapping.
- [4] Yesilmurat S., Isler V., “*Retrospective Adaptive Prefetching for Interactive Web GIS Applications*”, GeoInformatica 16(3): 435-466, 2012.
- [5] Park D. J., Kim H. J., “*Prefetch Policies for Large Objects in a Web-enabled GIS Application*”, Data & Knowledge Engineering, 37, 65-84, ISSN: 0169-023X, April 2001.
- [6] Funkhouser, T., “*Database Management for Interactive Display of Large Architectural Models*”, in Graphics Interface '96, ed. W. A. Davis and R. Bartels, (Canadian Human- Computer Communication Society, 1996, p. 1.
- [7] Aliaga D., Cohen J., Wilson A., Zhang H., Erikson C., Hoff K., Hudson T., Stürzlinger W., Baker E., Bastos R., Whitton M., Brooks F., Manocha D., “*MMR: An Interactive Massive Model Rendering System Using Geometric And Image-Based Acceleration*”, in: 1999 ACM Symposium on Interactive 3D Graphics, 1999, 199–206.
- [8] Kang Y. K., Kim K. C., Kim Y. S., “*Probability-Based Tile Pre-fetching and Cache Replacement Algorithms for Web Geographical Information Systems*”, Proceedings of the 20th International Conference on Advances in

- Geographic Information Systems, GIS '12, California, USA. New York: ACM, 349-358.
- [9] Li R., Guo R., Xu Z., Feng W., “*A Prefetching Model Based On Access Popularity For Geospatial Data in a Cluster-Based Caching System*”. International Journal of Geographical Information Science, 26(10): 1831-1844, 2012.
 - [10] Doherty, W. J., Thadani, A., J.: “*The Economic Value of Rapid Response Time*”, IBM. White Plains, NY, USA, 1982. Technical report GE20-0752-0.
 - [11] Brady, J., T., “*A Theory of Productivity in the Creative Process*”. In: IEEE Computer Graphics and Applications VI, 5, 25-34, 1986.
 - [12] Roast, C., “*Designing for Delay in Interactive Information Retrieval.*”, Interacting with Computers 10, 87-104, 1998.
 - [13] ZONA, “*The Economic Impact of Inaccessible Web Site Download Speeds*”, Zona Research Center, White Papers Zona Research Center, 1999.
 - [14] Lindstrom P. and Pascucci V. “*Terrain Simplification Simplified: A General Framework for View-Dependent Out-Of-Core Visualization*”. IEEE Transactions on Visualization and Computer Graphics, 239–254, 2002.
 - [15] Pascucci V. and Frank R. “*Global Static Indexing For Realtime Exploration of Very Large Regular Grids*”. Proceedings of the 2001 ACM/IEEE Conference on Supercomputing, 2001.
 - [16] García, R., de Castro J. P., Verdú M. J., Verdú E., Regueras L. M., and López P. “*An Adaptive Neural Network-Based Method for Tile Replacement in a Web Map Cache*”, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6782: 76-91, 2011.
 - [17] Jiang Y., Wu M.-Y., Shu W. “*Web Prefetching: Costs, Benefits and Performance*”, Proceedings of 7th International Workshop on Web Caching & Distribution, 2002.
 - [18] Quinn, S. and Gahegan, M. “*A predictive Model for Frequently Viewed Tiles in a Web Map*”. Transactions in GIS”, 14 (2), 193–216, 2010.

- [19] Kefaloukos, P. K., Salles, M. V., Zachariasen, M, “*TileHeat: A Framework for Tile Selection*”, ACM SIGSPATIAL GIS '12, November 6-9, 2012. Redondo Beach, CA, USA.
- [20] Varadhan G. and Manocha D. “*Out-of-Core Rendering of Massive Geometric Environments*”, Proceedings of the Conference on Visualization '02, pages 69–76, 2002.
- [21] Corrêa W. T., Klosowski J. T., Silva C. T. “*iWalk: Interactive Out-Of-Core Rendering of Large Models*”. Technical Report TR-653-02, Princeton University, 2002.
- [22] Rabinovich M. and Spatschek O., “*Web Caching and Replication*”, SIGMOD Record, 32(4):107–108, 2002.
- [23] Markatos, E., Chronaki, C., "A *Top-10 Approach to Prefetching on the Web*," Tech. Rpt. No. 173, Aug. 1996, ICS-FORTH, Heraklion Crete, Greece.
- [24] Tse, J. Altera Corp., El Cerrito, CA, USA, Smith, A. J., “*CPU Cache Prefetching: Timing Evaluation Of Hardware Implementations.*”, IEEE Transactions on Computers, 47(5), 509-526, 1998.
- [25] Han W., Whang K. Y., Moon Y. S., "A *Formal Framework for Prefetching Based on the Type-Level Access Pattern in Object-Relational DBMSs*", IEEE Transactions on Knowledge and Data Engineering, 17(10), 1436-1448, 2005.
- [26] Google Map Structure, <http://www.gisteam.de/ftp/Farbtafeln/76/76googleMapsStruc.pdf> , last visited on July 2014.
- [27] From Texture Virtualization to Massive Parallelization, http://mrl.cs.vsb.cz/people/gaura/agu/05-JP_id_Tech_5_Challenges.pdf, last visited on July 2014.
- [28] Lee, D., Kim, J., Kim, S., Kim, K., Yoo-Sung, K., & Park, J. , “*Adaptation of a Neighbor Selection Markov Chain for Prefetching Tiled Web GIS Data*”, Proceedings of the Second International Conference on Advances in Information Systems, vol. 2457, pp. 213-222. ISBN: 3-540-00009-7, 2002.
- [29] Podlipnig S., Boszormenyi L., “*A Survey of Web Cache Replacement*

- Strategies.*”, ACM Computing Surveys 35(4), 374–398, 2003.
- [30] Aggarwal, C., Wolf, J.L., Yu, P.S., “*Caching On the World Wide Web*”. IEEE Transactions on Knowledge and Data Engineering 11(1), 94–107, 1999.
 - [31] García R., de Castro J.P., Verdú E., Verdú M.J., Regueras L.M., “*An OLS Regression Model for Context Aware Tile Prefetching in A Web Map Cache*”, International Journal of Geographical Information Science, 2012.
 - [32] Fisher D., “The Impact of Hotmap”, <http://research.microsoft.com/apps/pubs/default.aspx?id=81244>, last visited on July 2014.
 - [33] Cox M. and Ellsworth D. “*Application-Controlled Demand Paging for Out-Of-Core Visualization*”. In Proceedings of the 8th Conference on Visualization ’97, pages 235–244, 1997.
 - [34] Doshi P. R., Rundensteiner E. A., and Ward M. O., “*Prefetching for Visual Data Exploration.*”, Eight Database Systems for Advanced Applications (DASFAA), 195-202. IEEE, 2003.
 - [35] Zadeh, L.A., “*Fuzzy Sets*”, Information and Control 8, 338–352, 1965.
 - [36] Bryan, L.A, Bryan E.A., “*Programmable Controller Theory and Implementation*”, Chapter Seventeen, pp. 798-845, 1997.
 - [37] Black, M., “Vagueness: An exercise in logical analysis”. Philosophy of Science 4: 427–455. Reprinted in R. Keefe, P. Smith (eds.): Vagueness: A Reader, MIT Press 1997, ISBN 978-0-262-61145-9.
 - [38] A Short Fuzzy Logic Tutorial, http://cs.bilkent.edu.tr/~zeynep/files/short_fuzzy_logic_tutorial.pdf, last visited on July 2014.
 - [39] Kulkarni, D. A., “*Chapter 3: Fuzzy Logic Fundamentals*”; Computer Vision and Fuzzy-Neural Systems; (Prentice Hall, 2007), 61-103.
 - [40] Ng C.-M., Nguyen C.-T., Tran D.-N., Tan T.-S., Yeow S.-W., “*Analyzing Prefetching in Large-Scale Visual Simulation*”. Proceedings of Computer Graphic International Conference, New York, NY, USA, 100–107, 2005.
 - [41] Ruffner, J. W., Lohrenz, M. C., & Trenchard, M. E., “*Human Factors Issues in the Development of an Advanced Digital Moving Map System*”. Proceedings

- of the Human Factors and Ergonomics Society Annual Meeting, 43(1), 31-35, SAGE Publications, 1999.
- [42] Lohrenz, M. C., Zuyle, P.V., Trenchard, M.E. Myrick, S.A., and Fechtig, S. D., “*Optimizing Cockpit Moving Map Displays for Enhanced Situational Awareness*”. Proceedings for the Symposium on Situational Awareness in the Tactical Air Environment. Dayton, OH: Crew Systems Ergonomics Information Analysis Center (CSERIAC), 1996.
 - [43] Product Focus: Moving Maps New Connections, http://www.aviationtoday.com/av/issue/feature/Product-Focus-Moving-Maps-New-Connections_737.html#.U7xf1_mSxBI, last visited on July 2014.
 - [44] Rogers, S. P., & Spiker, V.A., “*Computer-Generated Map Display for the Pilot/Vehicle Interface*”. SAE Transactions Journal of Aerospace, 97, 1.1148-1.1161, 1988.
 - [45] Unger, R. A., & Schopper, A. W., “*Digital Moving Map Displays For Fighter and Tactical Aircraft*” (CSERIACRA- 95-001). Wright Patterson Air Force Base, OH: Crew Systems Ergonomics Information Analysis Center, 1995.
 - [46] FliteScene® 2.7.3 Digital Moving Map Overview, http://download.harris.com/app/public_download.asp?fid=2389, last visited on July 2014.
 - [47] The Stratomaster Odyssey & Voyager, <http://www.lightflying.com.au/Stratomaster%20Pages/Odyssey.htm>, last visited on July 2014.
 - [48] FocusFlite Digital Moving Map, <http://focusflite.stm.com.tr/en>, last visited on July 2014.
 - [49] Domènech, J., Pont, A., Sahuquillo, J., Gil, J.A. “*An Experimental Framework for Testing Web Prefetching Techniques*”, Proceedings of the 30th EUROMICRO Conference, Rennes, France, 2004.
 - [50] Performance Specification of DTED, MIL-PRF-89020B, http://dds.cr.usgs.gov/srtm/version2_1/Documentation/MIL-PDF-89020B.pdf, last visited on July 2014.

- [51] Inertial Navigation System, http://en.wikipedia.org/wiki/Inertial_navigation_system, last visited on July 2014.
- [52] Global Positioning System, http://en.wikipedia.org/wiki/Global_Positioning_System, last visited on July 2014.
- [53] Android NDK, <http://developer.android.com/tools/sdk/ndk/index.html>, last visited on July 2014.

APPENDIX A

EAP FRAMEWORK

After having described EAP approach, now important components of framework with corresponding software counterparts are going to be examined through sub-sections. Although some of these components are already mentioned in previous section semantically, here their software counterparts in framework are explained. Beside, how can these components be used to extend and create new capabilities are going to be described.

As mentioned in various lines, the EAP is being used as framework so application that will utilize EAP employs it either through static library or dynamic libraries and manage it through provided services. In addition to this usage, EAP can also work as a client engine and provide all necessary data to multiple application through shared memory. It also accepts commands through a well-defined shared memory protocol.

A.1 EAP Packages

First of all, packages that contain the framework classes are described. These packages are depicted in Figure A-1.

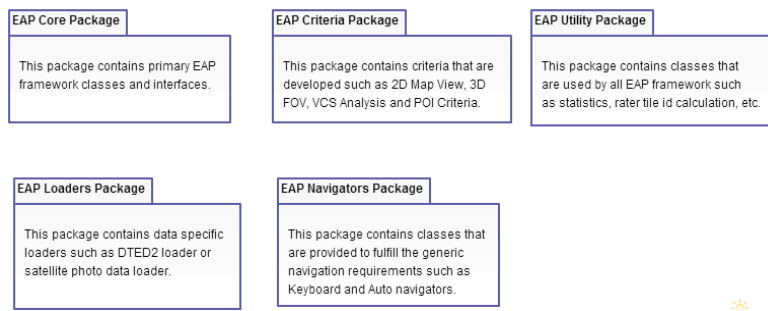


Figure A-1 Overview of VCS analysis criterion

There are five packages. The first one is the EAP core package which contains the core EAP framework classes which are EAPManager, RequestManager, CriteriaManager, ApplicationManager, NavigationManager and WBFISEngine. These classes form the EAP infrastructure and being used to perform EAP capabilities and operations. This package also contains interface class which are provided to be used by applications to derive new criterion, navigator and loader classes. The details of these classes and their relationship are described in the following lines.

The EAP criteria package contains the developed criteria and related classes which are described in chapter 5. Although, it is possible to develop new criterion by deriving from the interfaces provided with EAP core package, it would be much easier to use the basis criterion classes that are located in this package. These criteria are developed specifically for possible application capabilities like 2D map display, 3D terrain visualization, VCS analysis and etc.

The EAP loaders package contains the developed data loader and loader object classes. To be able to process the different kinds of data, EAP utilizes a plugin like mechanism which let applications register their data loader and loader objects through a configuration file mentioned in section 4.6 and appendix-B. Again the interface of these classes are provided in core package, this package contains the derived classes. The functions of these classes are described at class level in section A.2.

The EAP navigator package contains the developed keyboard and auto navigators that are used by applications to provide necessary criterion parameters like geographic position, altitude and speed. Normally, applications provide these to related criteria by taking them from corresponding sensors like Global Positioning System (GPS) or Inertial Navigation Systems (INS) [51, 52].

Finally, EAP utility package contains the supporting classes that are being used by all EAP framework such as statistics, unique tile identifier generator and briefing classes. These utilities can also be used by application that employs EAP.

A.2 EAP Classes

In this section, important classes that are developed for EAP framework are going to be described according to packages given in previous section. Some of these are already mentioned in different sections such as section 4.1 and section 4.6.2, here these are going to be described in more technical way.

A.2.1 EAP Core Package Classes

The overview of classes that EAP contains are shown in Figure A-2.

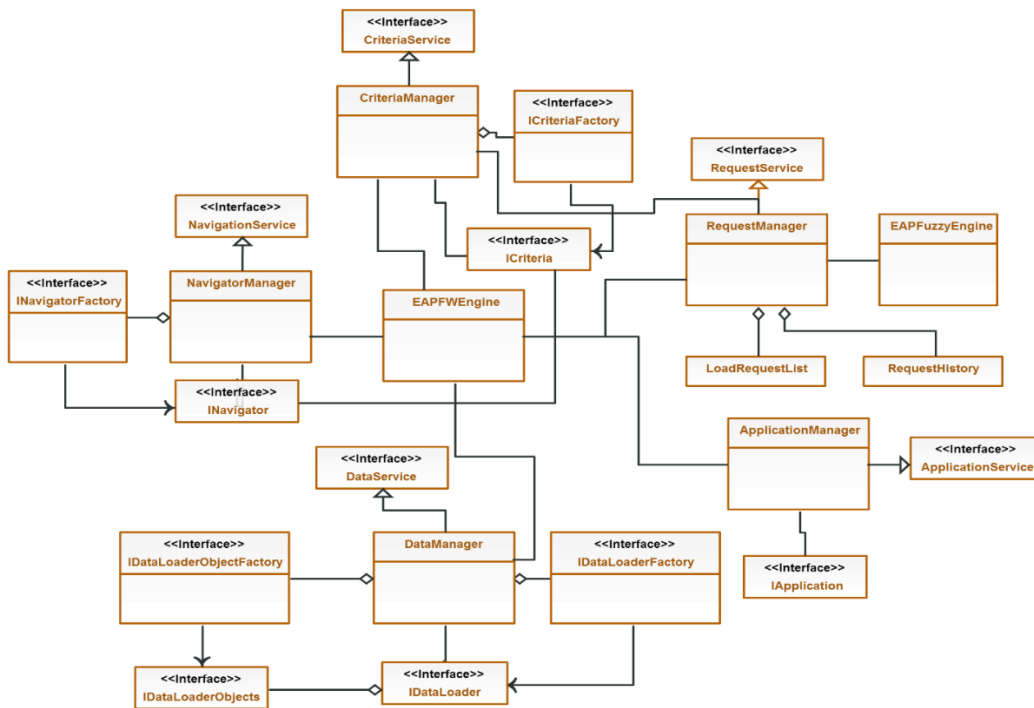


Figure A-2 Overview of EAP core package class diagram

All manager classes share same common life-cycle management methods like Configure, Initialize, Tick and Finalize. The Configure method passes the configuration file to manager classes and let these classes obtain the data from this file. The configuration and initialization are provided as two separate method to let all manager modules first obtain configuration file content and then perform initialization. Tick method is the main trigger method which is called periodically by EAPManager class with delta time. Finally, the Finalize method let manager classes perform any deallocations and free the resources allocated during execution.

- EAPFWEngine:

- This class represents the EAPManager that is illustrated in the architecture that is given in section 4.1. It is responsible from management of overall EAP framework and its interaction with application. It provides EAP services to application and vice versa. It also manages the other manager classes and calls the life cycle methods of each manager.

- **IApplication:**

- Represent the application from EAP perspective. It provides interface with methods to get application specific parameters. For instance, a digital moving map application should inherit this interface and provides an application class to EAP framework which can be used by registered criteria to query the current state such as current application display mode. One application is usually sufficient for most of the scenarios.

- **ApplicationManager:**

- This class is responsible from management of application classes registered and also passing the configuration file to registered application classes.

- **ICriterion:**

- This is the base interface class provided for all criteria and it provides necessary operations and services that all derived criteria require. Some of these operations are related with criterion life cycle such as configuration through provided XML configuration file, initialization and finalization. Some of these operations are related with EAP framework such requesting, prefetching, evaluation and replacement operations. Events are also provided. One such event is *requestCompleted* event which informs criterion about status of its request with corresponding information. There also events like pre/post-operations to let criteria perform operations before or after mentioned EAP operations. There are also methods to get the list of provided operations by criterion, to get its weight, name, and priority range and activation statuses.

- **CriteriaManager:**

- This class is responsible from management of registered criteria and their behavior. It is also illustrated and described briefly through the architecture that is given in section 4.1. It passes the evaluation of load and replacement requests to corresponding criteria. The weights of these criteria are also managed by this class and WB-FIE engine. This manager also provides necessary information about registered criteria to other framework classes and application. It also passes the active navigator to active criteria.

- **ICriteriaFactory:**
 - This interface let application to create criterion objects using factory pattern.
- **IDataLoaderObject:**
 - This is the lowest level of loader class interface which is responsible from loading and parsing operations. So application should derive this interface for each different raster data types such as raw or compressed or elevation or map and sources such as file system or file server that are going to be handled by EAP. Each loader object works as a different thread so that all loading operations are executed asynchronously. This class uses the raster metadata provided through configuration file for loading operations.
- **IDataLoaderObjectFactory:**
 - This interface let application to create data loader objects using factory pattern.
- **IDataLoader:**
 - This loader class is responsible from management of data loader objects mentioned above. It contains necessary management capabilities in itself and can be extended for specific purposes and extra operations. Passing tile load requests to corresponding loader objects, managing the loaded tile chunks, moving these chunks from cache to main memory, main memory to cache and disposing the not required tiles also performed by this class. This class also manages the allocation/deallocation of loaded tile chunks.
- **IDataLoaderFactory:**
 - This interface let application to create data loaders using factory pattern.
- **DataManager:**
 - This class represents the DataManager that is illustrated in the architecture that is given in section 4.1. This class is responsible from management of all data loaders and data loader objects. It provide necessary data loading services to other EAP framework classes. It also responsible from managing the raster data that is going to be used by system and provide necessary information about this data to other classes and application.
- **LoadRequest:**
 - This class holds data about the request that is initiated by a criterion like raster

tile id, initiator criterion, origin (prefetch or direct), data origin (local, remote), status (requested, loaded, load failed, etc), unique id (to identify each request), loader id, priorities and active requesters. This is the data that is used by most of the EAP classes and operations.

- **RequestManager:**

- This class represents the RequestManager that is illustrated in the architecture that is given in section 4.1. This class is mainly responsible from management of request initiated by criteria. It also performs passing load requests from CriteriaManager to DataManager and inform back about their statuses, load and replacement orderings, checking data manager for load requests, checking cache and initiate tile disposing if required. All loaded requests are analyzed by this class and then corresponding initiator is informed about the result of request. It accumulates all load and prefetch requests and forward them to corresponding criteria through Criteria Manager to come up with load order and similarly for replacement order in case of tile disposal. This manager also maintains historical information about each request.

- **INavigator:**

- An interface class which provide services that should be fulfilled by each navigator that is going to be used with EAP such as providing geographic location data, speed, altitude and etc.

- **INavigatorFactory:**

- This interface let application to create navigator objects using factory pattern.

- **NavigatorManager:**

- This class is responsible from management of navigators that are provided by applications. It also let application to choose an active navigator among these registered ones and provide it to CriteriaManager.

- **ICriteriaService, IDataService, INavigationService, IRequestService:**

- These service interfaces are being used to provide the capabilities and services that are mentioned above to each other framework classes and application.

A.2.2 EAP Criteria Package Classes

The overview of classes that EAP criteria package contains are shown in Figure A-3.

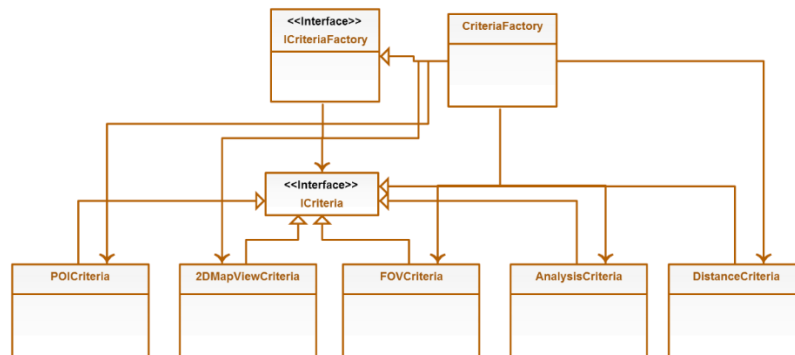


Figure A-3 Overview of EAP criteria package class diagram

- **2DMapViewCriteria:**
 - This class represents the developed criteria for 2D map displaying capability which is described in chapter 5 and is derived from ICriteria interface. It also implements the RAP capabilities and can switch one to other through provided enable method.
- **FOVCriteria:**
 - This class represents the developed criteria for 3D terrain visualization capability which is described in chapter 5 and is derived from ICriteria interface.
- **AnalysisCriteria:**
 - This class represents the developed criteria for vertical cross section analysis capability which is described in chapter 5 and is derived from ICriteria interface.
- **POICriteria:**
 - This class represents the developed criteria for point of interest display which is described in chapter 5 and is derived from ICriteria interface.
- **2DDistanceCriteria:**
 - This class represents the 2D distance evaluator criteria which is described in chapter 5 and is derived from ICriteria interface.

- CriterionFactory:

- This class is responsible from creating the criteria that are mentioned above using factory pattern and is derived from ICriteriaFactory interface.

A.2.3 EAP Loaders Package Classes

The overview of classes that EAP loaders package contains are shown in Figure A-4.

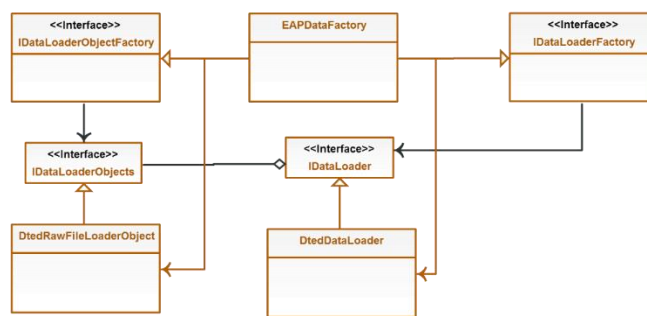


Figure A-4 Overview of EAP loaders package class diagram

- DtedRawFileLoaderObject:

- This class extends the IDataLoaderObject interface to load and parse raw DTED data which is stored as 512x512 tiles on file system.

- DtedDataLoader:

- This class extends the IDataLoader interface to load DTED data using DtedRawFileLoaderObject class instances.

- EAPDataFactory:

This class is responsible from creating data loader and data loader object that are mentioned above using factory pattern.

A.2.4 EAP Navigators Package Classes

The overview of classes that EAP navigators package contains are shown in Figure A-5.

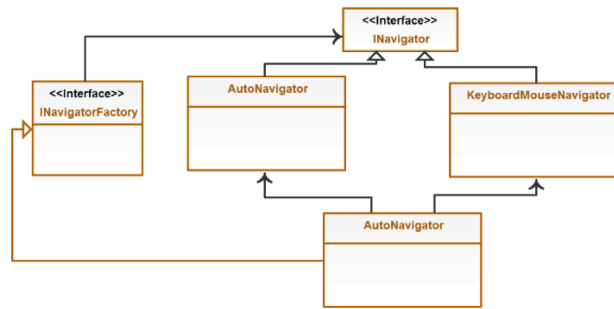


Figure A-5 Overview of EAP navigator package class diagram

- AutoNavigator:
 - This class extends the INavigator interface and being used to execute the pre-recorded or prepared flights as if they are currently being provided by application. This navigator can be considered as briefing tool. It is developed to run the EAP framework with prepared experimentation scenarios and recorded flights.
- KeyboardMouseNavigator:
 - This class extends the INavigator interface and mainly responsible from providing application inputs. For instance, keyboard is being used to simulate aircraft movement and mouse is being used to perform 2D navigation patterns such as zooming and panning.
- NavigatorFactory:
 - This class is responsible from creating navigators that are mentioned above using factory pattern.

A.3 EAP Utilization

As mentioned in contributions section, the EAP framework can also be used for testing and evaluation purposes. This is provided through criterion. For this purpose, developed prefetching mechanism should be ported to implement provided ICriteria interface which is described briefly above. Then it should be simply registered to CriteriaManager, the rest is managed by EAP engine. Moreover, the EAP framework can controlled through provided services.

The EAP framework is developed using C++ with considering cross platform usage so it can be used for both windows and android platforms using provided Android Native Development Kit (NDK) [53]. The framework is deployed as static library so applications that are utilize EAP only need to include and link this library.

APPENDIX B

AN EXAMPLE CONFIGURATION FILE

The configuration file is prepared in XML format and can be adapted for further usage. The words that are shown in “< >” are keywords that EAP are looking for. An example configuration file content is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Unique Ids should be started from 1 -->
<root>
  <Statistics>
    <!-- Type eUInt32 => 0, eInt32 => 1, eDecimal32 => 2 -->
    <StatisticType>
      <Id>0</Id>
      <Name>MissPredictCount</Name>
      <Type>0</Type>
    </StatisticType>
    <StatisticType>
      <Id>1</Id>
      <Name>CorrectPredictCount</Name>
      <Type>0</Type>
    </StatisticType>
    <StatisticType>
      <Id>2</Id>
      <Name>UnusedPredictionCount</Name>
      <Type>0</Type>
    </StatisticType>
    <StatisticType>
      <Id>3</Id>
      <Name>PredictCount</Name>
      <Type>0</Type>
    </StatisticType>
    <StatisticType>
      <Id>4</Id>
      <Name>RequestCount</Name>
      <Type>0</Type>
    </StatisticType>
    <StatisticType>
      <Id>5</Id>
      <Name>HitRatio</Name>
      <Label>HR</Label>
      <Type>2</Type>
    </StatisticType>
    <StatisticType>
      <Id>6</Id>
      <Name>UnusedPrectionRatio</Name>
      <Label>UPR</Label>
      <Type>2</Type>
    </StatisticType>
    <StatisticType>
      <Id>7</Id>
      <Name>CorrectPrectionRatio</Name>
```

```

    <Label>UPR</Label>
    <Type>2</Type>
  </StatisticType>
</StatisticType>
  <Id>8</Id>
  <Name>CriterionWeight</Name>
  <Label>Weight</Label>
  <Type>2</Type>
</StatisticType>
</StatisticType>
  <Id>9</Id>
  <Name>CriterionActivityLevel</Name>
  <Label>CAL</Label>
  <Type>2</Type>
</StatisticType>
</StatisticType>
  <Id>10</Id>
  <Name>FuzzyEngineOperation</Name>
  <Label>CAL</Label>
  <Type>0</Type>
</StatisticType>
</Statistics>
<DataManagement>
  <SharedMemoryLabel>CAPDataMngmntSharedMemory</SharedMemoryLabel>
</DataManagement>
<DataLayers>
  <!-- System wise data layer information -->
  <DataLayer Name="DTED2 Elevation Data" LayerUniqueID ="1">
    <LoaderName>ED2Loader</LoaderName>
    <DataType>Raster</DataType>
    <Desc>This layer contains 30m resolution elevation data which is provided through tiles.</Desc>
    <CommonRasterProperties>
      <BoundCount>1</BoundCount>
      <ResolutionX>0.00027777800000000023</ResolutionX>
      <ResolutionY>0.00027777799999999996</ResolutionY>
    </CommonRasterProperties>
    <!-- Each data layer may contain different data sources in different formats like RAW, ECW -->
    <DataSource DataSourceID="1">
      <Desc>This source contain raw data dted2 data.</Desc>
      <DataFormat>RAW</DataFormat>
      <DataRootPath>../Data/RasterData/</DataRootPath>
      <FileNameTemplate>ED2</FileNameTemplate>
      <IsTiled>1</IsTiled>
      <TileSize>512</TileSize>
      <DataSourceRasterProperties>
        <Width>9001</Width>
        <Height>10801</Height>
        <PixelFormat>short</PixelFormat>
        <PixelSize>2</PixelSize>
        <PyramidCount>0</PyramidCount>
        <NoDataValue>-9999.0</NoDataValue>
        <ExtentCenterX>35.7500009999999997</ExtentCenterX>
        <ExtentCenterY>39.4999988000000000</ExtentCenterY>
        <ExtentWidth>2.50027977800000021</ExtentWidth>
        <ExtentHeight>3.0002801779999997</ExtentHeight>
      </DataSourceRasterProperties>
    </DataSource>
    <DataSource DataSourceID="2">
      <Desc>This source contain ecw dted2 data.</Desc>
      <DataFormat>ECW</DataFormat>
      <DataRootPath>../Data/DTED2Data/</DataRootPath>

```

```

    <FileNameTemplate>ED2ECW</FileNameTemplate>
    <IsTiled>>false</IsTiled>
    <DataSourceRasterProperties>
        <ExtentCenterX>35.50000999999997</ExtentCenterX>
        <ExtentCenterY>39.0</ExtentCenterY>
        <ExtentWidth>19.0</ExtentWidth>
        <ExtentHeight>6.0</ExtentHeight>
    </DataSourceRasterProperties>
</DataSource>
</DataLayer>
</DataLayers>
<DataLoaders>
    <!-- For each layer there will be one data loader -->
    <!-- These loader objects should be registered before configuration is started!-->
    <DataLoader Name="ED2Loader" LoaderUniqueID="1">
        <Desc>This loader will be used to load RAW DTED2 Elevation Data.</Desc>
        <DataLayerUniqueIDToLoad>1</DataLayerUniqueIDToLoad>
        <LoaderObjectName>ED2RawFileLoaderObject</LoaderObjectName>
        <LoaderObjectCount>2</LoaderObjectCount>
        <!-- Overall memory allocation -->
        <MaxAllocatedChunkSize>2512</MaxAllocatedChunkSize>
        <!-- Memory portion reserved for cache -->
        <CachedChunkSize>64</CachedChunkSize>
        <NoOfCacheItemsToDispose>8</NoOfCacheItemsToDispose>
        <!-- eAllocatedAsRequest => 0, eAllocateInitially => 1-->
        <AllocationMethod>0</AllocationMethod>
        <SharedMemoryLabel>ED2LoaderSharedMemory</SharedMemoryLabel>
    </DataLoader>
</DataLoaders>
<DataLoaderObjects>
    <!-- There might be different loader objects which are responsible from loading data from different
        sources or different formats of same data layer-->
    <DataLoaderObject Name="ED2RawFileLoaderObject" DataLoaderObjUniqueID="1">
        <Desc>This object is responsible from loading RAW DTED2 elevation data from file system.</Desc>
        <DataSourceIDToLoad>1</DataSourceIDToLoad>
        <!-- The wait in msec before each load op. For debug purposes -->
        <WaitPeriod>0</WaitPeriod>
    </DataLoaderObject >
    <DataLoaderObject Name="ED2ECWFileLoaderObject" DataLoaderObjUniqueID="2">
        <Desc>This object is responsible from loading ECW DTED2 elevation data from file system.</Desc>
        <DataSourceIDToLoad>2</DataSourceIDToLoad>
        <!-- The wait in msec before each load op. For debug purposes -->
        <WaitPeriod>0</WaitPeriod>
    </DataLoaderObject >
</DataLoaderObjects>
<CriteriaManagement>
    <!-- This element will be used to configure general criteria management parameters like adaptive or direct
        loading, replacement policy, priority ranges, etc -->
    <IsAdaptive>1</IsAdaptive>
    <!--Unique ids of registered set of criteria-->
    <ActiveCriteriaList>
        <Criteria Id ="1"/>
        <Criteria Id ="2"/>
        <Criteria Id ="3"/>
    </ActiveCriteriaList>
</CriteriaManagement>
<RequestManagement>
    <PrefetchLoadSlotSize>10</PrefetchLoadSlotSize>
</RequestManagement>
<CriterionList>
    <!-- For each criterion that will be used in system there will be one item -->

```

```

<!-- Both generic and criterion specific parameters will be defined here -->
<Criterion Name="2DMapView" CriterionUniqueID="1">
  <!--Common attributes-->
  <Desc>This criterion mainly deals with 2D navigation operations like panning and zooming. </Desc>
  <HasRequestCapability> 1</HasRequestCapability>
  <HasEvaluateCapability>1</HasEvaluateCapability>
  <HasPrefetchCapability> 1</HasPrefetchCapability>
  <ActivateRequestCapability>0</ActivateRequestCapability>
  <ActivateEvaluateCapability>0</ActivateEvaluateCapability>
  <ActivatePrefetchCapability>0</ActivatePrefetchCapability>
  <ActivateReplacementCapability>0</ActivateReplacementCapability>
  <HasReplacementCapability> 1</HasReplacementCapability>
  <EmployStatistics>1</EmployStatistics>
  <InitialWeight>0.5</InitialWeight>
  <MinWeight>0.05</MinWeight> <!--These values require reasoning!-->
  <MaxWeight>0.95</MaxWeight>
  <MinPriorityRange>0.0</MinPriorityRange>
  <MaxPriorityRange>1.0</MaxPriorityRange>
  <AffectedBySpeed>1</AffectedBySpeed>
  <RegisteredDataLoaders>
    <Raster LoaderName="ED2Loader"/>
    <!--1 => Dted2-->
  </RegisteredDataLoaders>
  <!--Specific attributes-->
  <!--This extent parameters are given in degree -->
  <MapViewExtentWidth>0.4</MapViewExtentWidth>
  <MapViewExtentHeight>0.3</MapViewExtentHeight>
  <MapViewScale>500000</MapViewScale>
  <NumberOfRowsToBePrefetched>1</NumberOfRowsToBePrefetched>
  <!--This max range is given in nm -->
  <MaxPrioritizationRange>200.0</MaxPrioritizationRange>
  <!--If this flag is set then this map view is shown as rectangle and not fit into original map view-->
  <FitIntoScreen>0</FitIntoScreen>
  <!--This threshold values are given in percentage the percentage of a single tile with/height in default-->
  <LatitudeThreshold>50.0</LatitudeThreshold>
  <LongitudeThreshold>50.0</LongitudeThreshold>
</Criterion>
<Criterion Name="Analysis" CriterionUniqueID="2">
  <!--Common attributes-->
  <Desc>This criterion mainly perform vertical cross section analysis.</Desc>
  <HasRequestCapability> 1</HasRequestCapability>
  <HasEvaluateCapability>1</HasEvaluateCapability>
  <HasPrefetchCapability> 1</HasPrefetchCapability>
  <HasReplacementCapability> 1</HasReplacementCapability>
  <ActivateRequestCapability>0</ActivateRequestCapability>
  <ActivateEvaluateCapability>0</ActivateEvaluateCapability>
  <ActivatePrefetchCapability>0</ActivatePrefetchCapability>
  <InitialWeight>0.5</InitialWeight>
  <EmployStatistics>1</EmployStatistics>
  <MinWeight>0.05</MinWeight>
  <!--These values require reasoning!-->
  <MaxWeight>0.95</MaxWeight>
  <MinPriorityRange>0.0</MinPriorityRange>
  <MaxPriorityRange>1.0</MaxPriorityRange>
  <AffectedBySpeed>0</AffectedBySpeed>
  <RegisteredDataLoaders>
    <Raster LoaderName="ED2Loader"/>
    <!--1 => Dted2-->
  </RegisteredDataLoaders>
  <!--Specific attributes-->
  <InitialAnalysisLength>30</InitialAnalysisLength>

```

```

<AnalysisPrefetchLength>5</AnalysisPrefetchLength>
<!--This extent parameters are given in degree -->
<MaxPrioritizationRange>200.0</MaxPrioritizationRange>
<!--If this flag is set then this map view is shown as rectangle and not fit into original map view-->
<FitIntoScreen>0</FitIntoScreen>
<!--This threshold values are given in percentage-->
<LatitudeThreshold>30.0</LatitudeThreshold>
<LongitudeThreshold>30.0</LongitudeThreshold>
</Criterion>
<Criterion Name="FOV" CriterionUniqueID="3">
  <!--Common attributes-->
  <Desc>This criterion mainly deals with 3D visibility based navigation operations.</Desc>
  <HasRequestCapability> 1</HasRequestCapability>
  <HasEvaluateCapability>1</HasEvaluateCapability>
  <HasPrefetchCapability> 1</HasPrefetchCapability>
  <ActivateEvaluateCapability>0</ActivateEvaluateCapability>
  <ActivateRequestCapability>0</ActivateRequestCapability>
  <ActivatePrefetchCapability>0</ActivatePrefetchCapability>
  <ActivateReplacementCapability>0</ActivateReplacementCapability>
  <HasReplacementCapability> 1</HasReplacementCapability>
  <EmployStatistics>1</EmployStatistics>
  <InitialWeight>0.5</InitialWeight>
  <MinWeight>0.05</MinWeight>
  <!--These values require reasoning!-->
  <MaxWeight>0.95</MaxWeight>
  <MinPriorityRange>0.0</MinPriorityRange>
  <MaxPriorityRange>1.0</MaxPriorityRange>
  <AffectedBySpeed>1</AffectedBySpeed>
  <RegisteredDataLoaders>
    <Raster LoaderName="ED2Loader"/>
    <!--1 => Dted2-->
  </RegisteredDataLoaders>
  <!--Specific attributes-->
  <!--This parameters are given in degree -->
  <InitialFOVYaw>90.0</InitialFOVYaw>
  <InitialFOVPitch>0.0</InitialFOVPitch>
  <InitialFOVRoll>0.0</InitialFOVRoll>
  <InitialFOV>60.0</InitialFOV>
  <FOVDepth>25.0</FOVDepth>
  <!--This parameters are given in nm -->
  <NearPlane>0.01</NearPlane>
  <FarPlane>15.0</FarPlane>
  <!--This max range is given in nm -->
  <MaxPrioritizationRange>100.0</MaxPrioritizationRange>
  <!--This threshold values are given in degrees-->
  <FOVThreshold>2.5</FOVThreshold>
  <BearingThreshold>2.5</BearingThreshold>
  <!--This threshold values are given in percentage the percentage of a single tile with/height in default-->
  <LatitudeThreshold>50.0</LatitudeThreshold>
  <LongitudeThreshold>50.0</LongitudeThreshold>
</Criterion>
<Criterion Name="POI" CriterionUniqueID="4">
  <!--Common attributes-->
  <Desc>This criterion deals with POI data that increase the priority of corresponding raster tiles.</Desc>
  <HasRequestCapability> 1</HasRequestCapability>
  <HasEvaluateCapability>1</HasEvaluateCapability>
  <HasPrefetchCapability> 1</HasPrefetchCapability>
  <HasReplacementCapability> 1</HasReplacementCapability>
  <EmployStatistics>1</EmployStatistics>
  <Initial2DWeight>0.5</Initial2DWeight>
  <Initial3DWeight>0.5</Initial3DWeight>

```

```

<Min2DWeight>0.0</Min2DWeight> <!--These values require reasoning!-->
<Max2DWeight>0.7</Max2DWeight>
<Min3DWeight>0.0</Min3DWeight>
<Max3DWeight>0.7</Max3DWeight>
<MinPriorityRange>0.0</MinPriorityRange>
<MaxPriorityRange>1.0</MaxPriorityRange>
<AffectedBySpeed>0</AffectedBySpeed>
<RegisteredDataLoaders>
  <Raster>ED2Loader</Raster>
  <!--1 => Dted2-->
</RegisteredDataLoaders>
<!--Specific attributes-->
<!--This threshold values are given in percentage where this will be used according to radius of POIs
    i.e. if poi interest radius is 10 nm than request will be 11 for %110, and 15 nm for %150-->
<RequestDistance>110</RequestDistance>
<PrefetchDistance>150</PrefetchDistance>
</Criterion>
</CriterionList>
<NavigatorManagement>
  <ActiveNavigator>2</ActiveNavigator>
</NavigatorManagement>
<NavigatorList>
  <Navigator Name="KeyboardMouseNavigator" NavigatorUniqueID="1">
    <Desc>This navigator provides user to navigate using only keyboard. The moving/panning/rotation/scale
up/down are provided through shortcuts</Desc>
    <InitialLatitude>39.2</InitialLatitude>
    <InitialLongitude>35.6</InitialLongitude>
    <!--How much will navigator move in each tick in degree-->
    <DegreePerKeyStoke>0.000005</DegreePerKeyStoke>
    <!--How much will navigator rotate bearing in each tick in degree-->
    <RotateDegreePerKeyStoke>0.001</RotateDegreePerKeyStoke>
  </Navigator>
  <Navigator Name="AutoNavigator" NavigatorUniqueID="2">
    <Desc>This navigator provides necessary capabilities for experimentation.</Desc>
    <InitialLatitude>39.2</InitialLatitude>
    <InitialLongitude>35.6</InitialLongitude>
  </Navigator>
</NavigatorList>
</root>

```

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Uluat, Mehmet Fatih
Nationality: Turkish (TC)
Date and Place of Birth : 26 August 1981, Van
Marital Status : Married and have one children.
Phone : +90 535 629 06 22
Email : f_uluat@yahoo.com
tr.linkedin.com/pub/mehmet-fatih-uluat/64/423/69

EDUCATION

Degree	Institution	Year of Graduation - CPGA
M. Sc.	Dept. of Computer Engineering, METU	2007 - 3.43 / 4
B. S.	Dept. of Computer Engineering, METU	2004 - 3.27 / 4
High School	Van Private Serhat Science College	1999 - 4.95 / 5

WORK EXPERIENCE

Year	Place	Enrollment
2011-Present	STM, A.Ş.	Digital Map Application Group Leader
2008-2011	Simsoft	Project Manager
2004-2008	ASELSAN	Expert Software Engineer
2003-2004	Meteksan Sistem	Software Engineer

FOREIGN LANGUAGES

Advanced English

PUBLICATIONS

Journal Publications and Book Chapters

1. Uluat, M. F., İşler V.: “A Fuzzy Logic Based Ensemble Adaptive Tile Prefetching”, IJGIS, International Journal of Geographic Information Science, 2014, (Submitted and revision is received).
2. Uluat, M.F., Oguztüzün, H.: “Model Based Approach to the Federation Object Model Independence Problem.”, In ISCIS (2011), 451-459.

Conference Publications

1. “UML Applications in Real-Time Systems”, published and presented in National Software Engineering Symposium 2005 (UYMS), as an author with Ersel Ercek

and Sevdâ Erdoğdu, Ankara, Published in II. Ulusal Yazılım Mühendisliği Sempozyumu, 2005.

2. “Pedestal Mounted Air Defense Missile System Training Simulator”, published and presented in National Defense Application Modeling and Simulation Conference 2005(USMOS), as an author with Ersel Ercek, Ankara (UYMS 2005).

CERTIFICATIONS

- HTML 5 technology,
- Beden Dili Eğitimi,
- Real-Time Design Patterns,
- Software Architecture Design For Large Scale and Complex Systems,
- Object Oriented Software Development With Case Tool Rhapsody 6.0,
- MSDN TRAINING: Developing Microsoft .Net Applications For Windows in C#,
- High Level Architecture,
- Anlayarak Hızlı Okuma.