AEGIS - A state-of-the art component based spatio-temporal framework for education and research

Roberto Giachetta

Dept. of Software Technology and Methodology,
Eötvös Loránd University,
Budapest, Hungary

Abstract

In past years, geoinformation has gained a significant role in information technology due to the spread of GPS localization, navigation systems and the publication of geographical data via Internet. The inclusion of semantic information and temporal alteration has also become increasingly important in GIS. The overwhelming amount of spatial and spatio-temporal data resulted in increased research effort on processing algorithms and efficient data management solutions.

This article presents the AEGIS framework, a currently developed spatio-temporal data management system at the Eötvös Loránd University, Faculty of Informatics (ELTE IK). This framework will serve as the future platform of GIS education and research at ELTE IK. It aims to introduce a data model for the uniform representation of raster and vector data with temporal references; to enable efficient data management using specialized indexing; and to support internal revision control management of editing operations. The framework offers a data processing engine that automatically transforms operations for distributed execution using GPGPUs, allows fast operations even with large datasets and high scalability with regard to new methods.

To demonstrate the usage of the system two prototype application – segment-based image classification and agent-based traffic simulation – are also presented.

Keywords

Geospatial Information Systems, spatio-temporal data, indexing data structures, GPU based data processing, revision controls of data, remotely sensed image clas-

sification, agent based traffic simulation.

1 Introduction

Geographical information systems (GIS) have undergone a spectacular development in the past years. Beside traditional areas of GIS applications a rapid development has taken place in the world of navigation systems as well. Google Maps and NASA World Wind, together with their Application Programming Interface (API), are common tools in the global handling of spatial data. The world of open source software has also evolved a lot. There is a rising need for professionals whose practice cover both information technology and geography. This paradigm shift has to be taken into account both by professionals and by academic people.

At the Eötvös Loránd University, Faculty of Informatics (ELTE IK) the informal association, called Creative University GIS Workshop (TEAM) deals with several related research topics, e.g. Intelligent Raster image Interpretation System (IRIS), University Digital Map Library (EDIT), Virtual Globes Museum (VGM) and segment-based analysis of remote sensing images. An important collaboration takes place both in education and research with the Institute of Geodesy, Cartography and Remote Sensing (FÖMI) . This governmental institution is responsible for the research, development and application of remote sensing in Hungary, mainly in the areas of agriculture and environmental protection.

Based upon the experience gained with research and education, the plan of a standalone geographic framework, called AEGIS has been outlined. It is designed for broad functionality, efficiency, and allowing students to skip the learning curve and concentrate on their main tasks in lab projects and thesis works without the need of building up auxiliary functionality from scratch. The system is based on standards of the Open Geospatial Consortium (OGC)[1], and is not intended to be a competitor of current industrial GIS solutions or GIS toolkits such as GeoTools[2], but rather to be a prototype and sandbox for experimental research and education.

Students are offered step-by-step learning of standards, algorithms and solutions, easy to undestand, documented source code, and well structured components that can be used for many types of GIS projects, including thesis works. Researchers may also find many possibilites in this flexible, yet stable environment to test results and compare them with existing solutions. Even the framework itself has several experimental features that are research topics, including multi-level spatio-temporal indexing solutions, revision control of spatial data and automated GPU based data processing.

[1] `http://www.opengeospatial.org`

[2] `http://www.geotools.org`

The AEGIS concept has been introduced in Giachetta et al. (2012), and is currently under development. Some features are only outlined in the paper to be implemented in the future. The framework is to be released as open-source project in the future.

The rest of the paper is arranged as follows. In Section 2 we will outline the concept and architecture of our system. Section 3 explains details of the AEGIS data management model. Section 4 presents the data processing capabilities and solutions. Section 5 describes the first applications of the system, namely segment based image classification and agent-based traffic simulation. Section 6 concludes the paper.

2 System architecture

AEGIS has component based architecture in order to be as adaptable and customizable as possible. Several class libraries are stacked together to build up system functionality and relevant libraries can be chosen for any application ranging from desktop editors through mobile viewers to server services. These libraries also provide API functionality for building custom applications. The AEGIS framework stack contains the following components (see Figure 1).

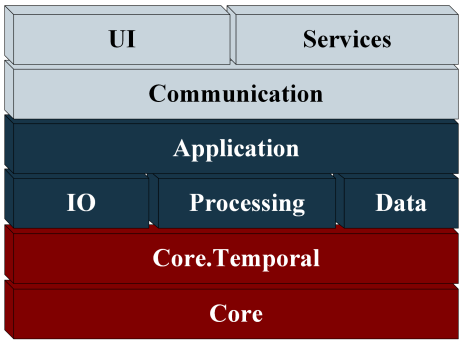


Figure 1: AEGIS framework stack

- *Core* is the heart of the framework containing the OGC and ISO compliant 2D/3D spatial data model and reference model including several utility classes (e.g. indexing structures). This component is designed not to contain any temporal references as not all GIS applications require it. However base interfaces are provided for time management.
- *Core.Temporal* is the extension of the data model with temporal properties, reference model and spatio-temporal indexing structures supporting multiple

time measurement solutions. Our solution focuses less on moving objects and trajectory and more on modeling stationary spatial features with limited temporal validity, for instance, road sections closed due to constructions and remote sensing image series of crop growth. Details of the model can be read in Section 3.

- *IO* contains the import/export possibilities for GIS file formats and network services. Several standard spatial formats are supported. The module allows the consumption of OGC web services including WMS and WFS. Support for further services is planned.

- *Data* component is responsible for database management. All spatio-temporal data imported into the system can be stored in a database. AEGIS is designed to handle spatial and spatio-temporal information (including indexing, revision control, etc.) entirely within the system, hence no spatial features of the database system are required. It is clear that this solutions lacks the effectiveness and potential offered by most database management systems. However it provides researches a stable high level environment to work in. Based on our earlier experience MongoDB[3] has been chosen as the primary database backend.

- *Processing* offers spatio-temporal vector/raster processing operations. For both the Processing and the IO components functions are defined using a meta descriptor system to support runtime management and extension. Processing services also offer an internal programming scheme based on expression trees. By using this scheme the processing instructions can easily be transformed to low level GPU instructions and automatically distributed. An internal processing engine is responsible for this purpose. Details of processing can be read in Section 4.

- *Application* layer is responsible for user authentification, editing, processing and import/export management including the runtime handling of available IO and Processing functions. During feature editing changes are overseen and logged to enable revision control of the data. The layer also contains a scripting engine using a dedicated high level domain specific language to enable the creation of custom processing operations and batch processing of data even from multiple sources and formats.

- *Communication* layer contains services for internal communication between application frontend (User Interface and Services) and backend (Application

[3] `http://www.mongodb.org`

layer). The primary objective of the layer is to enable the separation of these two layers and the creation of thin clients that can execute operations on remote servers. Communication uses a service oriented architecture with encryption and user authentification.

- *Services* is responsible for providing web services, mainly in OGC standard formats (WMS, WPS, etc.). This server side module is not to be confused with the client side IO component which offers OGC service clients to enable import of data into the system.

- *UI* provides graphical user interface components and 2D/3D data visualization methods for multiple platforms including desktop and mobile environments. There are also plans for extending visualization using augmented reality on mobile devices.

Using the component based design, the structure of the system can be configured with each intallment. For instance, a thin client installation of AEGIS may only require Core, UI and Communication components as data storage, processing, import and export are performed by the server which is connected to the client via Communication layer. In contrast the server does not require UI and IO modules, but should include Services to enable access to data for other applications. A thick client may contain all components except Services and can support local or remote execution of processes. Figure 2 shows this layout possibility.

The implementation is carried out using the Microsoft .NET Framework, because of the wide possibilities and the simple usage of this development platform, the strong .NET education carried out at ELTE and also the success of .NET based GIS software products such as DotSpatial[4] and SharpMap[5].

In the current phase of development the higher layers (Application, Communication, Services and UI) are only partially complete. Additional research, design and testing will follow before these components are implemented.

3 Spatio-temporal data management

Modeling and indexing of spatio-temporal objects has been a frequent topic among researchers since the 90s, and many data models have been introduced (Abraham & Roddick 1999, Nguyen-Dinh et al. 2010). However for spatio-temporal data neither common solutions have arisen, nor standards have been developed so far. In contrast, spatial data has well known standards maintained by OGC.

[4] `http://dotspatial.codeplex.com`
[5] `http://sharpmap.codeplex.com`

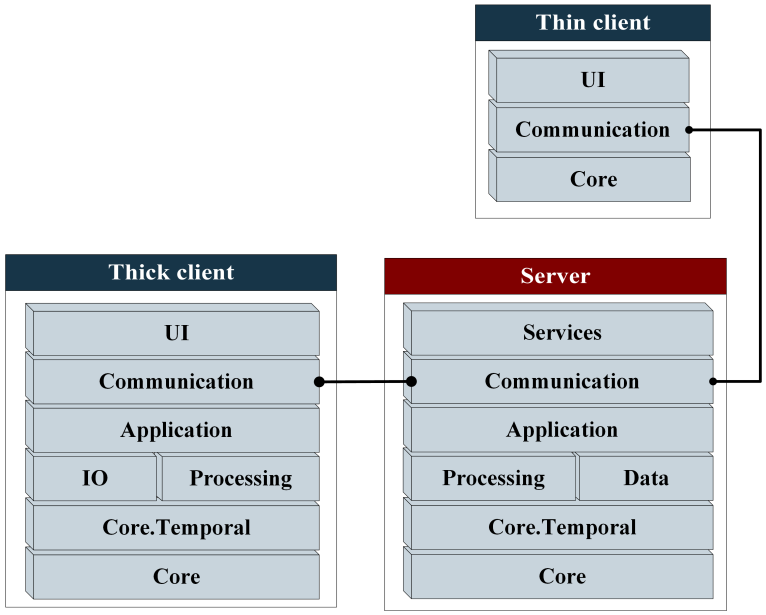


Figure 2: Example layout of AEGIS components with thin and thick clients

As explained in Section 2 the AEGIS data model is implemented in two components.

- *Core* holds the spatial model including raster and vector data representation, spatial indexing structures and reference system management. Basic interfaces are defined for temporal relations but are not implemented in the component. The usage of Core components is recommended in applications where no temporal properties are needed. For all spatial vector features the representation is a direct mapping of the OGC Simple Feature Access standard (Herring, J. R., ed. 2011). However the standard has been extended with several new types, including raster support. Detailed description of this model follows in Section 3.1. The reference model is an implementation of the OGC Spatial Referencing by Coordinates specification (Cooper, P., ed. 2010) using EPSG guidelines[6].

- *Core.Temporal* extends the core model with temporal properties and references. Hence base classes and interfaces are imported from the Core package, the spatio-temporal model can function as strictly spatial if needed.

[6] `http://www.epsg.org/guides/`

Besides adding temporal properties to features spatio-temporal indexing is introduced. Temporal references are managed with multiple temporal reference systems (e.g. Clocks, Calendars). The reference model is a partial implementation of *ISO 19108:2002 Geographic information – Temporal schema* (2002).

3.1 Unified vector and raster data representation

In the OGC SFA data model the central item is the *Geometry* class, which has several specialized versions (including collections) in object inheritance taxonomy. Geometry defines the interface for spatial properties and operations among any spatial objects. The model focuses on two and three dimensional vector data without any temporal references. In our implementation of this standard, multiple extensions have been introduced to enable flexible and effective implementation of features (see Figure 3).

Vector and raster data are required to be handled in the same manner so that operations work uniformly on them. In current geospatial systems the representation and operations of raster and vector data are usually independent of each other. In our system, raster data is introduced as descendand of Geometry. All spatial operations, which are defined on vector objects (e.g. intersection, translation, projection etc.), can also be performed on raster data (for example the ability to intersect an orthophoto with a land parcel polygon). By using the meta descriptor system restrictions can be applied to any processing operation to prevent raster functions (e.g., intensity transformations, image filtering) working on vector data.

For raster data the *Raster* and *RasterBand* classes are introduced. RasterBand is a descendant of Rectangle and contains one band of a raster image, while Raster is a collection of multiple bands of an image. These classes are enhanced with operations and properties related to raster imagery. Images can be stored in several radiometric resolutions (from 8 bit to 64 bit for every band), may contain pyramid layers, and image masks (to indicate actual image pixels).

Vector geometries can be transformed to graph representation using the *GeometryGraph* class that contains points as edges and paths as vertices. This representation enables merging multiple geometries to form a single network. Equivalently the *RasterGraph* class is the graph representation of raster data where vertices contain intensity informations of pixels. Graphs may include a multi-level structure, where a single high level vertex contains multiple low level vertices. This is useful for example, managing road hierarchies or raster image segments.

New collections, called maps have been introduced primary for efficiently managing large amount of objects. Map classes are build up using R-tree based indexing structures for fast spatio-temporal query of objects, as described in Section 3.3.

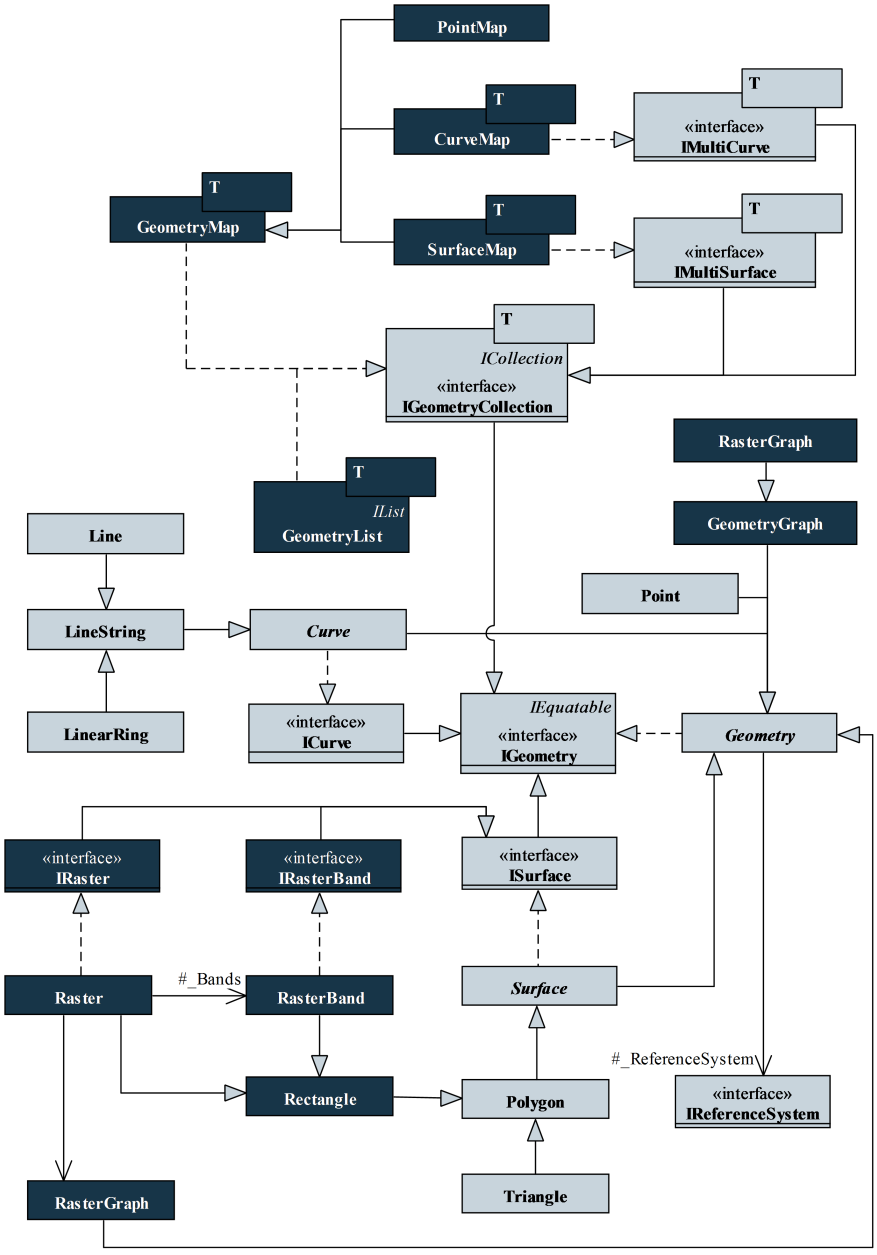


Figure 3: The data model based on Simple Feature Access

All geometries have been extended by the possibility of storing the time instant or time interval in which they are considered to be valid. Generally, items in a collection do not need to exist in the same time interval, but restrictions can be made to force temporal equality among all items of the collection. The reference system of these temporal geometries is usually a compound reference system containing a temporal system to manage calendar or clock aspects.

All features may contain any number of descriptive data (metadata) stored as key/value pairs. All geometries within a collection require to have the same metadata schema. The metadata is also time-dependent and is only valid when the feature is valid, but values can change multiple times within the validity. These changes may or may not depend on the change of the feature's spatial properties.

It must be noted that in the current phase of development not all operations work as efficiently as other implementations of the SFA standard (e.g., GeoTools, DotSpatial) as clarity, readability and stucture of the source code is among top priorities, but steady progress is expected also in this factor.

3.2 Data storage

The data model has been specifically designed to handle all spatial and temporal properties of the data at system level without requiring any spatial support from the database system. Hence the database backend has been chosen based on previous experimental results on performance of queries and modifications (Máriás et al. 2012). Hence, currently MongoDB serves this purpose. In the future support is planned for PostGIS and other database systems as well.

MongoDB is a document-oriented database management system (DODBMS), which enables the schema-less storage of hierarchical data (Padhy et al. 2011). Due to the large variety of spatio-temporal and descriptive data it has proven to be more efficient in data querying than SQL based systems. MongoDB has only minimal spatial support for 2D geometry, but the structure of the stored items can be easily customized, even to allow the storage of different types of geometries in a single collection. Thus, no spatial operations function for these geometries in the DODBMS, but still, simple spatial and temporal queries (such as bounding box based selection) are usable.

For storing raster data the internal *GridFS* convention is used that is specifically designed for storing large amount of binary information linked to the document. Again it can be noted that no image operations function directly at database level.

It can be clearly seen that out solutions lacks the speed and potential of spatial database operations, but has multiple gains. The AEGIS data model can be aesily mapped to the document-based structure of MongoDB, even raster data by using GridFS. Extending the database management system with new data models and

indexing solutions would be far more complex than building the required infrastructure at system level, so research in these areas can progress quicker, whereas the results can be adopted into database systems in the future.

3.3 Spatio-temporal indexing of geometries and metadata

As described in Section 3.1, there are multiple temporal aspects treated in the data model.

- Features have limited temporal validity, a time interval in which they are considered to be valid. This interval may be the same for the entire collection, but can be different for each object.
- Features may change spatial properties during their validity. In the case of moving objects, the entire geometry is relocated, but is not changed. However our model enables any change in the geometry itself, even a single point may alter within a polygon leaving the rest in place.
- Metadata may change within the validity. Metadata keys and values can change independently from each other and from spatial alteration. The values indentified by the same key but with different timestamp can be independent or in relation to each other. For instance the average travel speed of a road section for a given timestamp can be expressed as a factor of the speed limit.

To support this broad spectrum of temporal management, indexing has been introduced on multiple levels. For spatio-temporal indexing *3D/4D R-trees* (Kolovson & Stonebraker 1991) and *Multiversion 3D/4D R-trees* (MV3R-trees) are used (Tao & Papadias 2001). In both cases, time is the last (third or fourth) dimension. The former is efficient for timestamp based queries, the letter has good performance in both timestamp and interval queries. Both trees have spatial and temporal bounding boxes, and use multiple heuristics to enhance the performance of tree updates. For temporal indexing of metadata, *Multiversion B-trees* (MVB-trees) are included (Kolovson & Stonebraker 1991).

Using these structures the following multi-level indexing solution was implemented, as illustrated in Figure 4).

- Map collections contain MV3R-trees to support both interval and timestamp queries.
- Features contain 3D/4D R-trees to access spatial properties at the specified timestamp.

- Metadata collections of the features use MVB-tree to query values at the specified timestamp.

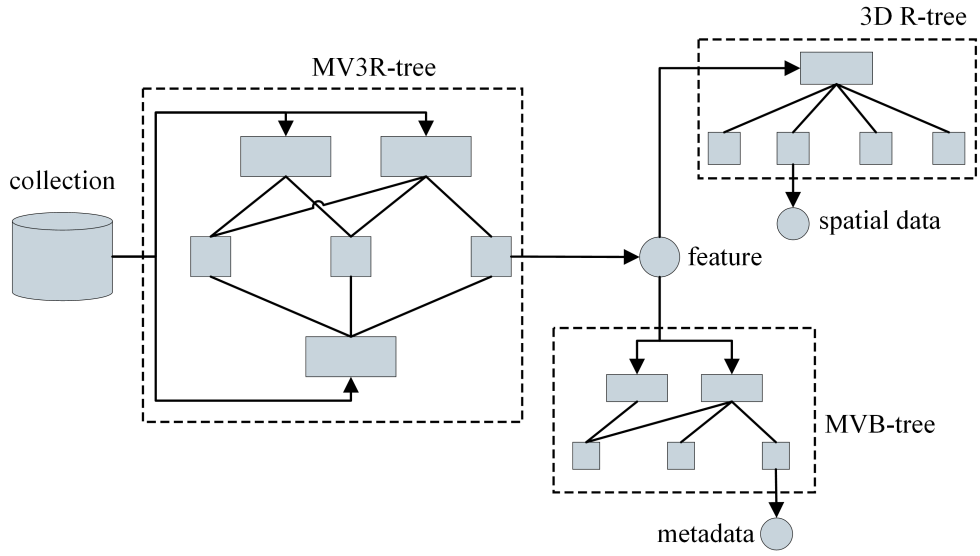


Figure 4: The multi-level indexing solution

In certain situations some features can belong to multiple collections, and have different temporal properties in these collections. For example, a road section can belong to a collection containing daily traffic information (where average travel speeds are calculated for every hour), and also to a collection with yearly traffic information (where average travel speeds are calculated for every day). Different metadata values are need to be calculated in the two collections. In the simplest solution this would require the duplication of the feature.

To prevent this duplication, additional B+-trees have been included in collections and graphs. This can be seen in Figure 5. The solution is specifically designed for situations where relations can be established between the altering metadata. The tree contains timestamps as keys, and for each timestamp, a collection of temporal variables. These variables contain descriptive information that changes in time, and is not stored in the feature. Variables are formed by (key, modificiation, usage) triplets. These triplets define metadata variability for any descriptive value of the geometry. Key refers to property name, usage defines application of the modification, such as override, add, multiply, etc.

For instance, the speed limit may be stored as metadata in the feature, whilst the tree contains the average travel speed for a given hour as a factor of the speed limit.

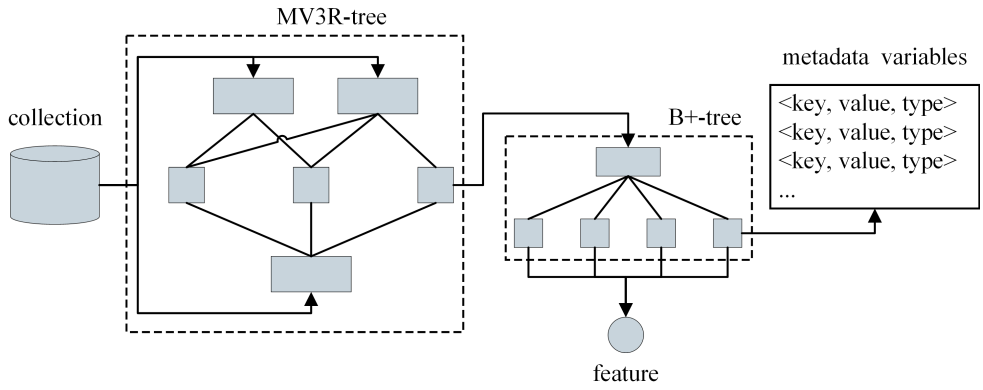


Figure 5: The MV3R-tree extended with B+-trees

When querying the speed for a certain timestamp, the speed limit is multiplied by the factor providing the temporal variability.

The metadata variability is only reachable through the collection, and different collections provide different alterations of the metadata values. However this solution is only effective when the amount of changes and therefore the space required by the B+-trees is less than the space required for the feature duplication. Since collections may cointain other collections, this effectiveness can be raised by grouping features into inner collections, where the value change factors are the same.

3.4 Revision control of data

Revision control is an essential part of modern application development. Hence it is also usable in any field where the modifications of documents are needed to be tracked. In this regard spatio-temporal features may be considered as documents where editing operation cause changes in either spatial, temporal or metadata properties of the feature. In vector representation editing operations are restricted on spatial and temporal geometry properties but in raster representation they may also be changes involving pixel data. Only a handful of studies deal with revision control of images. Modern revision control solutions offer only limited possibilities for managing binary data.

Our solution, based on Chen et al. (2011), is integrated as a part of the framework. The core idea is the storage of image operations using a directed acyclic graph (DAG) where each vertex contains a performed editing operation. This representation is suitable for nonlinear revision control and does not require the storing

of the entire modified image.

In the AEGIS system, all editing operations on geometries are executed as either vector based modifications (including all raster image changes) or as simple variable changes (such as metadata modification). Therefore it is fully compliant with the graph based storage. The method is also applicable for editing vector features. However in our system a graph is maintained for the entire editing lifetime of a feature, not just between two revisions. Therefore all previous revisions are obtainable and the framework only needs to store the original raster and the head revision of each branch (if needed). Revisions may also be tagged. A tagged revision stores the entire geometry, so it can be retrieved without any need of reapplying editing operations.

The revision control process is fully automated and reacts to all changes signaled by the feature that reach the Application level.

4 Flexible data processing

Data processing, algorithmic capabilities and supported import/export formats are the key questions in most applications and services. Not only the number and purpose of processing algorithms should be taken into account but also their performance is crucial. All these aspects were taken into account during the design of AEGIS with the inclusion of dynamic extendibility and GPGPU based distributed execution.

Dynamic extension is supported by using reflection capabilities of the .NET framework and the meta descriptor service, that is used by the Processing and IO packages. Algorithms may be implemented in any .NET language using the AEGIS API, which offers many base classes for file or remote service access and algorithm description. These compiled class libraries can be simply added to the system dictionaries to be automatically recognized and usable through UI commands and the scripting engine.

The scripting engine offers an easy to learn high level procedural language, currently named *AEGIScript*. It is possible to create batch scripts on any import/export and geometry processing operation; to combine simple algorithms to form composite, reusable operations and to execute algorithms sequentially and in parallel.

The efficient execution of processing algorithms is another issue. It is widely known that intermediate language programs executing on virtual machines cannot match the speed of native applications. Relying strictly on .NET code in algorithm execution would result to poor performance. Hence algorithms can be written as expressions, which are dynamically compiled and transformed to either .NET intermediate language (IL) or OpenCL language codes. In the latter both CPU and

GPU based executions are supported to enhance processing speeds. Thus the programmer may write the algorithm in simple .NET code, which can be executed as low level code on parallel GPGPU architecture. The complete structure of the processing environment can be seen in Figure 6.

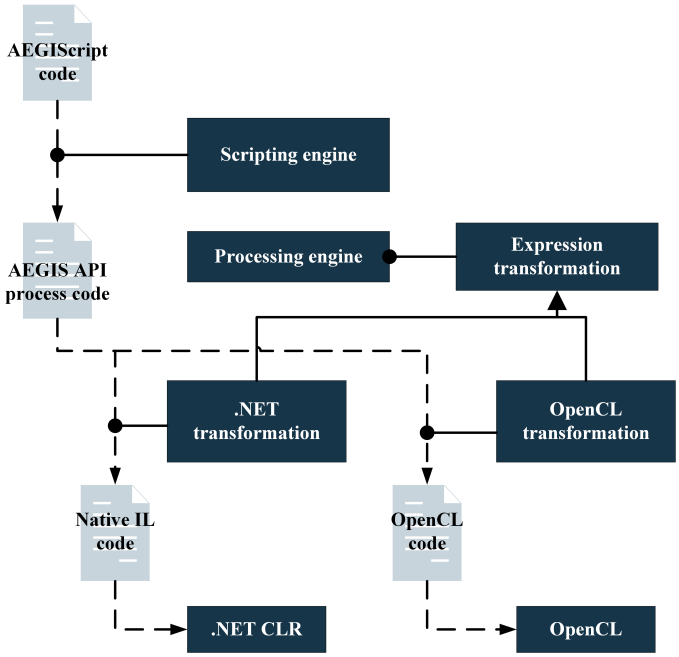


Figure 6: Conrtol flow of the processing services

This compilation is performed by the processing engine. The solution is effective in handling of large raster and vector data. However input data must be first converted to the specified simple format of the execution environment (e.g., simple one dimensional arrays for OpenCL) and back after the operation. The transformation is automated, but can be time-consuming. After transformation, algoritms can work either in place or out place on the data (as defined by the meta descriptor of the process).

The processing engine is also extendible with new execution environments including other GPU based models (such as CUDA) or other platforms (e.g., database system). It must be noted that in the current phase of development not all processing operations are supported by the transformation engine, these are executed as standard .NET code.

5 Applications

The first application, based on AEGIS architecture, is a simplified agent-based suburban traffic simulation. The simulation calculates and displays traffic information, mainly congestion levels for every time of the day within a year. The main goal of the application is the testing of the implemented indexing solutions with vector data. The application has been been developed using the Core, Core.Temporal and IO packages of the AEGIS stack; separate operations and display environment are built on top.

The second application is a segment based classification library for remote sensing images. The algorithms have been previously implemented in both C++ and C#. Hence, the goal of the application is the comparision of performance and efficiency of the Processing component and the OpenCL transformation.

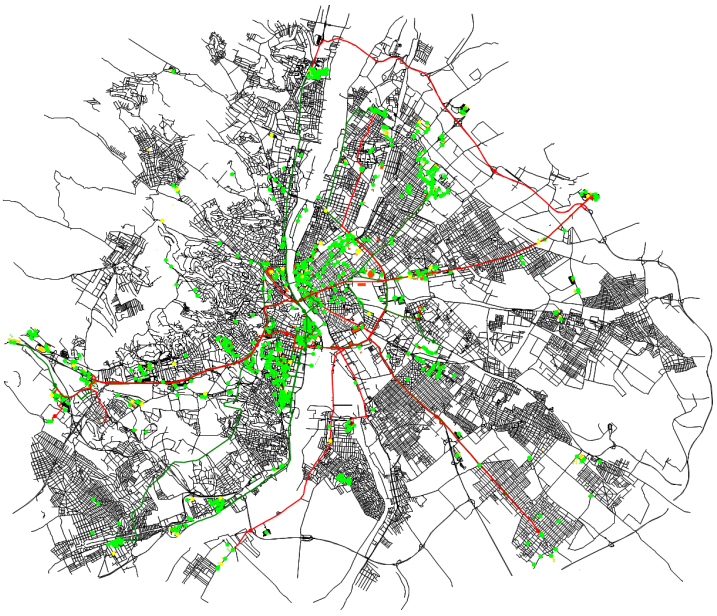
Both applications are under development, therefore results are preliminary. Definitive results will be published in the future.

5.1 Agent based traffic simulation

In this simulation a number of independent agents drive on the road nework during the day. The agents have multiple destination addresses to drive to. Two main destinations are workplace and home; but multiple random targets (like shopping centers or restaurants) may also occur. Agents travel to all locations by car. Locations are built up using metadata of building objects in the city map. Agents present in this simulation are primitive; they use simple distribution-based algorithms to make decisions. For example, an agent may have a working time from 8 o'clock in the morning to 6 in the afternoon, and can go to multiple stores or other locations after work.

Figure 7 illustrates the visualization of the traffic simulation on the map of Budapest, Hungary. In Figure 7a agents traveling to destination are shown in red, agents travelling home in green. Figure 7b displays the traffic congestion levels (yellow for small congestion, red for large congestion) for 9:00 AM of a specified day.

Agents plan their routes using *A*-algorithm* working on the *GeometryGraph* representation of the road network. Routing algorithm calculates the journey based on road travel time, which is available as metadata. The calculation is performed beforehand, based on historical traffic information combined from yearly and daily data including the last results prior to calculation. During travel the actual driving speeds are calculated based on the number of agents and the loadability of the road section. The traffic data is recalculated every 15 minutes. Due to traffic jams occuring, the agents speed can become too low or the travel time can become too



(a) Running agents

(b) Traffic congestion levels

Figure 7: Visualization of the agent based traffic simulation

high causing the agent to performe one of multiple actions, including recalculation of the entire route avoiding nearby road sections, recalculation of partial route, or even altering the destination address.

To manage the traffic data, only a single copy of every road section is used for all calculations by multiple graphs in combination with B+-trees for metadata variability storage. This solution can be seen in Figure 8. Data is summarized and optimized (including rebuilding of B+-trees) after each day. Measurements show that although optimization of the collections is quite time consuming, the query of values is only 15-20% less efficient than using multiple instances of the road sections, causing 5-8% performance loss of the entire simulation, with the benefit of using only 55-60% of storage space compared to multiple instances.

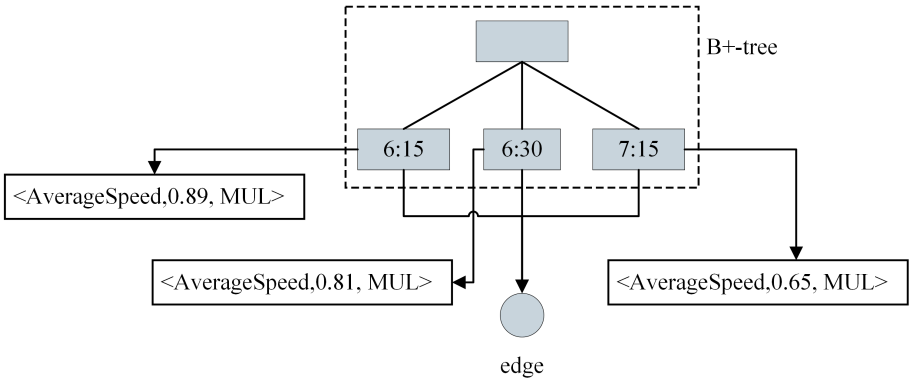


Figure 8: Temporal metadata indexing used by the traffic simulation

The application also served for inspecting the efficiency of the MV3R-tree by blocking road sections and so limiting their temporal validity, but no significant performance differences were measured in case of acceptable amount of changes.

5.2 Segment based image classification

To achieve the proper detection of features in mapping, image objects, that fit land cover objects, are needed to be delineated and they must be classified into predefined classes. However, traditional pixel-based classification methods completely disregard spatial relations.

In recent years, scientists of ELTE and FÖMI have been jointly investigating efficient segmentation methods of remote sensing images. Up to now the implementation and investigation of several algorithms have taken place, including the analysis of parameterization and accuracy (Dezső, Giachetta, László & Fekete 2012).

For testing and comparison purposes, several dedicated applications and libraries have been developed and some open-source and proprietary products have been used including *eCognition* (Dezső, Fekete, Gera, Giachetta & László 2012). However these solutions lack of flexibility and customization possibilities which are needed for accurate results. As a result implementation of several segmentation methods and segmentation based classification algorithms have begun recently in the AEGIS framework.

These approaches include both merge-based and cut-based methods, namely *sequential linking*, *best merge* and *minimum mean cut*. All algorithms work on the graph form of the image, represented by the RasterGraph class, that can store either pixel or segment information in vertices. These graphs are easily convertible to arrays that can be handled by OpenCL. By defining the steps of the algorithm through expressions, the entire process can run on GPU after the transformation of the graph with the same results as in previous implementations.

Initial results are promising regarding execution time. Using OpenCL based processing, the performance increase is up to 70-75% compared to standard .NET implementation and 30-40% compared to previous C++ implementation. However, further investigation and optimization will take place before trustworthy results can be presented.

6 Conclusion and future work

In the previous sections the concept of the AEGIS spatio-temporal framework and the goals of the authors' research have been introduced. Although development is still in early stage and several aims are still in planning phase, some results have already been reached.

The system is based on the spatio-temporal data model as described in Section 3, which uses complex data structures and multi-level indexing with temporal variability to enable more efficient management of data. Processing services allow flexible extensions and dynamic transformation of algorithms to GPGPU execution for improved performance. Processing capabilities are extended by the inclusion of batch scripting.

Further research includes the testing of other indexing structures and representation possibilities, and their performance measurement using more applications; the enhancement of algorithm transformation for wider support on different operation types and the implementation of higher levels in the AEGIS infrastructure, namely UI, Communication and Services.

References

Abraham, T. & Roddick, J. F. (1999), 'Survey of Spatio-Temporal Databases', *GeoInformatica* **3**(1), 61–99.
URL: *http://dx.doi.org/10.1023/A:1009800916313*

Chen, H.-T., Wei, L.-Y. & Chang, C.-F. (2011), 'Nonlinear revision control for images', *ACM Trans. Graph.* **30**(4), 105:1–105:10.
URL: *http://doi.acm.org/10.1145/2010324.1965000*

Cooper, P., ed. (2010), *The OpenGIS Abstract Specification - Topic 2: Spatial referencing by coordinates, version 4.0*, Open Geospatial Consortium.

Dezső, B., Fekete, I., Gera, D. A., Giachetta, R. & László, I. (2012), 'Object-based image analysis in remote sensing applications using various segmentation techniques', *Annales Universitatis Scientiarum Budapestinensis de Rolando Eotvos Nominatae Sectio Computatorica* **37**, 103–120.
URL: *http://ac.inf.elte.hu/Vol_037_2012/103_37.pdf*

Dezső, B., Giachetta, R., László, I. & Fekete, I. (2012), Experimental study on graph-based image segmentation methods in the classification of satellite images, *in* R. Vaughan, ed., 'EARSel eProceedings', Vol. 11, pp. 12–24.
URL: *http://www.eproceedings.org/static/vol11_1/11_1_laszlo1.html*

Giachetta, R., László, I. & Bálint, C. L. (2012), Towards the new open source GIS platform AEGIS, *in* A. Degbelo, J. Brink, C. Stasch, M. Chipofya, T. Gerkensmeyer, M. I. Humayun, J. Wang, K. Broelemann, D. Wang, M. Eppe & J. H. Lee, eds, 'GI Zeitgeist 2012 - Proceedings of the Young Researchers Forum on Geographic Information Science', Heidelberg: Akademische Verlagsanstalt, pp. 11–22.

Herring, J. R., ed. (2011), *OpenGIS Implementation Standard for Geographic Information: Simple Feature Access - Common Architecture, version 1.2.1*, Open Geospatial Consortium.
URL: *http://www.opengeospatial.org/standards/sfa*

ISO 19108:2002 Geographic information – Temporal schema (2002), International Organization for Standardization, Geneva, Switzerland.
URL: *http://www.isotc211.org/Outreach/ISO_TC_211_Standards_Guide.pdf*

Kolovson, C. P. & Stonebraker, M. (1991), 'Segment indexes: dynamic indexing techniques for multi-dimensional interval data', *SIGMOD Rec.* **20**(2), 138–147.
URL: *http://doi.acm.org/10.1145/119995.115807*

Máriás, Z., Nikovits, T., Takács, T. & Giachetta, R. (2012), 'A study of storing large amount of inhomogeneous data in workflow management systems', *Annales Universitatis Scientiarum Budapestinensis de Rolando Eotvos Nominatae Sectio Computatorica* **37**, 275–292.
URL: *http://ac.inf.elte.hu/Vol_037_2012/275_37.pdf*

Nguyen-Dinh, L.-V., Aref, W. G. & Mokbel, M. F. (2010), 'Spatio-Temporal Access Methods: Part 2 (2003 - 2010)', *IEEE Data Eng. Bull.* **33**(2), 46–55.

Padhy, R. P., Patra, M. R. & Satapathy, S. C. (2011), 'RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database's', *International Journal of Advanced Engineering Sciences and Technology* **11**(1), 15–30.
URL: *http://ijaest.iserp.org/ijaest-archieves-vol11.1.html*

Tao, Y. & Papadias, D. (2001), MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries, *in* 'Proceedings of the 27th International Conference on Very Large Data Bases', VLDB '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 431–440.
URL: *http://dl.acm.org/citation.cfm?id=645927.672363*