Frame Paradigm and Object-Oriented Image Processing for Photogrammetry

Yuri V. Vizilter, Yuri V. Morzeev, Alexander A. Stepanov, Sergei Yu. Zheltov

State Research Institute of Aviation Systems (GosNIIAS)

Vicktorenko, 7, Moscow, 125167, Russia

tel. (095) 157.97.48, fax. (095) 157.50.97

e-mail: yury@fenix.niias.msk.su

ABSTRACT

The Complete Object-Oriented approach is generalized for the photogrammetry-oriented application development. The advantages of the object-oriented data representation and management are discussed. It is shown that the object-oriented programming is very attractive for *data simulation, data unification* and *fractal data representation* in the digital photogrammetry software design. The original *frame paradigm* for object-oriented data management is described. The *structured processing model* is introduced to define the application area of this conception. It is proved that any full photogrammetric processing cycle satisfies this model.

Keywords: object-oriented programming, data representation, data management, semantic frames.

1. INTRODUCTION

Any method and category of thinking in every concrete field of modern science are determined by its "language," in generally, i.e. by the customary terminology. The origin and the fast development of the digital Image Processing (IP) and Digital Photogrammetry (DP) were caused by and then closely connected with the progress in the electronic computers' development. In terms and concepts of IP and DP we can obviously see the influence of the computer programming (for example, the IP algorithms are named as "IP procedures"). At the foundation of the digital image processing ideology, the FORTRAN was the standard in programming. Since that time, programmers had some great revolutions in their minds. The last and most important revolution was connected with the *Object-Oriented approach* (OOA) agreement. Due to this, some image processing engineers implement some object-oriented features in their program applications. For example, different object oriented languages are proposed for Image Algebra realization (e.g. [1]).

In the work [2] we have proclaimed a set of some basic principles for the image processing system development named as *Complete Object-Oriented Processing* (COOP). The essence of this COOP approach is the following. When you start the development of any new data processing software, you must make the decisions about four principal positions. They are: *data representation, algorithm implementation, processing management* and *user's interface* of the system. The problem that to be solved is what kind of the program structures will be the best for each of these four tasks. **Table 1** demonstrates the traditional answers and the answers that the object-oriented programming (OOP) give for each these four points. It is clear, that today we just can't see the pure traditional systems. Most of modern data processing systems are mixed. For example, they often have a nice graphic user's interface that is usually object-oriented. Some systems use the objects for data representation. But only when you select objects as an answer for all these four questions, you may call your system "complete object-oriented".

COOP parts*:*	Traditional approach	OOA
Data Representation	Arrays, Structures, Lists, ...	Unified Data Objects
Algorithms descriptions	Procedures, Functions	Semantic Objects
Processing Management	Command Language	Frame Object Net
User Interface	Command Line Interface	Graphic Window Interface

Table 1. Comparison of traditional and object-oriented approaches.

This COOP approach was proposed for Image Algebra implementation in the general image processing context. However, proposed principles are usable for any special data processing area. The important task is to determine what object-oriented aspects and features will be the most important and useful in the certain problem area. Concerning the four outlined COOP aspects, this task does not require to take in account two of them. The object-oriented algorithm representation is a conception that connected more with general philosophy of the algorithm development than with the certain features of the proper problem area. And the object-oriented user's interface is a common point of nowadays and needs no additional arguments to use it in any problem-oriented system.

We shall consider the object-oriented data representation and management applying to digital photogrammetry in this paper . The benefits of OOP in the photogrammetric data representation will be briefly discussed and the Frame Paradigm that realizes the OOA in the data management will be described more detail.

2. OBJECT-ORIENTED DATA REPRESENTATION

Image processing systems use many various data structures such as arrays, lists, records, etc. For example, images are usually represented as two-dimensional arrays of integers. All modern programming languages support such structures. However, programmers can have many problems even with these simple structures. The simple case of the possible problems is the following. The architecture of *Intel x86 processors* determines the segmentation of the PC RAM into the segments of size 64Kbyte. This limit was the reason of some special structure design to simulate the large 2D-arrays. One popular way was to define the 1D-array of pointers, where each pointer points to 1D-array (image row). When you want to operate with such structure as with a usual 2D-array, you must realize it as a C++ class and redefine the [•]-operator for this class. Many programmers many years use the objects for data representation in the manner like this. One can refer such technique as "object-oriented **data simulation**" (OODS). In the most broad sense, the OODC contains any object-oriented tools that provide more comfortable data access.

In the digital photogrammetry, especially in the low cast PC-based systems, the image data simulation and operation are the great problems because of necessity of operating with "super large" data arrays (more than PC hard disk memory). This problem and our OODC implementation for PC photogrammetry system is described in [3].

Besides data simulation, we have, at least, two another really important reasons to represent our DP-data as an objects. First of them is the following. In the image processing, there are many data types that are logically equivalent but have a different "physical" realization. Thus, data type *Image* can be: bynary image (one bit/pixel), intensity image (4-12 bit/pixel), color image (RGB, YQL,...; 16-32 bit/pixel), etc. Moreover, often it is convenient to interpret some processing results as the *Image* data type. For example, filtered images, gradient fields, depth maps and so on. "Physically", all these structures are the 2D-arrays, but the elements of these arrays are different: unsigned and singed integers of several bit lengths, float numbers, vectors, etc. Today we know only the one effective way how to use all these different structures in the common manner - object-oriented programming. We must define the basic abstract class *TImage* and construct all other required classes as its' heirs. In this case we can call all common features of these objects as the *TImage* features using the ***polymorphism*** of the classes. Evolving this idea, we can build the big ***inheritance tree*** (or inheritance wood may be) based on the principle of

grouping data types with some common logical features. This approach is useful for all data types, not for image data only. We refer this reason to use the OOP as "**data unification**".

The second our reason is connected with the well-known **fractal** nature of the image data. It is the feature of the image autosimilarity. It means that any small part of the image is similar (in the some sense) to the whole image. Many authors note the great importance of this image's feature but up to now we have only the one effective fractal application in the image processing area, namely, the fractal dimension measurement for the texture segmentation. In our opinion, there is a very interest possibility to use the fractal properties in the photogrammetric image analysis based on the object-oriented approach.

The traditional way to use the image autosimilarity is the *pyramid* processing based on the simple *scaling* idea (Fig. 1). It's obvious, that scaling expresses only the *dimensional* but not the *structural* autosimilarity of the image. The ordinary pyramid is the hierarchical data structure where the corresponding elements of the closest pyramid levels are connected with the logical links of "father-son" type. It is well known that the pyramid data representation provides an effective implementation for all matching techniques that are of use in the DP. For example, the *constraint matching* (Gruen, 1985) uses the Gaussian pyramid and the *contour-based stereo matching* (e.g. Marr, 1979) uses the Laplassian pyramid.

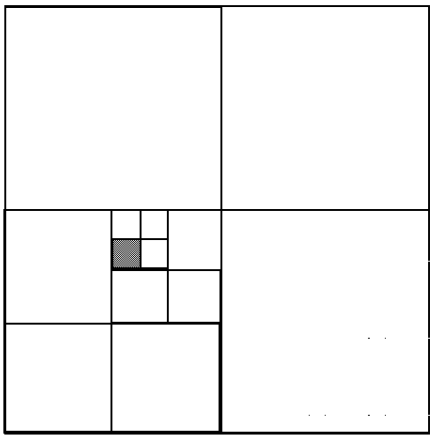

Fig. 1. Image "segmentation" provided by the usual image pyramid.

Let's consider the usual Gaussian pyramid with a scale parameter *s*=2. In the object-oriented manner, such pyramid can be represented as a set of the similar objects, where the "vertex" object of the pyramid contains the references to **n**=4..16 "daughter" objects and so on up to the lowest level of the pyramid where the objects contains references to the pixels of the image (Fig. 2). With this structure the traditional "top-down" and "down-top" pyramid processing implemented through the *messages exchange* between the objects of the different pyramid levels. It is still the usual pyramid. However, the ***polymorphism*** allows to build the *fractal pyramid* that has the *different* objects in its structure but still process them in the old uniform way. For example, different objects may have different numbers of the interlevel links and different rules of processing of the messages that received from the "fathers", "sons" and "neighbors". So, the process of building of such fractal pyramid data representation, at the same time, is the process of the fast primary image analysis. For instance, when we use the fractal Laplassian pyramid in the contour-based matching, it will be a "structure-to-structure" matching instead of the usual "contour-to-contour" technique. Moreover, the fractal representation provides the much more flexibility and many other new interest possibilities for the image processing algorithm developers. For example, it is very perspective for the "active vision" conception implementation.

It seems that the fractal pyramid representation will be of use for a depth map representation too, for example, in the scene decoding and recognition. However, we haven't explored this possibility yet.

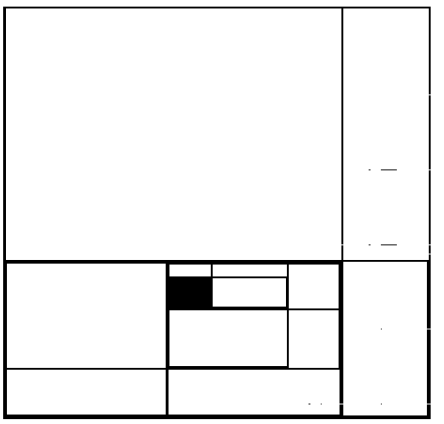

Fig. 2. Image "segmentation" provided by the *fractal* image pyramid.

Thus, we can say that the OOP is very attractive for *data simulation, data unification* and *fractal representation* in the digital photogrammetry software design.

3. DATA MANAGEMENT USING FRAMES

Some years ago the problem of processing management has a unique and simple solution: the operator types the *command* and the corresponding system activity takes place. Of course, one can create the *command file*, that contains the set of system commands, and then run this file to execute the required process automatically. Moreover, we can mask this command language by the set of icons and say that we use a *visual programming.* It is still the system that *controlled by commands.*

It was a revolution in the PC world when the command based operation system DOS was improved by the Microsoft Windows, the system that is *controlled by events*. However, it was only the first step from the command approach to the object-oriented approach because all of these system events are initiated by the user's activity. The ***primarity of data*** is one of the main principles of the OOP. The true data primarity means that the objects of the system generate the system events automatically according to the occurred changes in the system data and these events control the following data processing.

When we discuss the problem of data management applying to the proper problem area, the main question is whether the future system really requires some special management tools or not. One can note that any processing software with a powerful subsystem of data management has the following important properties:

(a) the processing schemes must consist of the set of some elements (procedures, operators);
(b) there is a natural sequence of the procedure calls because the outputs of some procedures are the inputs of the other procedures;
(c) different sensitive processing schemes are available through the combination of the system elements.

Each of these properties is essential. If the processing scheme is complex but not segmented into some independent elements then the corresponding data management contains only the input/output description. If all of data flows in the system are parallel then all system events will be only of the "start" and "stop" types. Finally, if the system realizes only the one processing scheme then this scheme can be implemented as an entire procedure that need no data management, i.e. the reduction to the case (a) takes place. Similarly, if the system realize some different processing schemes but all of them contain only the original elements without intersections between different schemes then the reduction to the case (b) takes place. The processing model that satisfies three mentioned conditions (a), (b) and (c) can be referred as a **structured processing model** (SPM). For example, it is easy to see that any image algebra-based system realizes the kind of SPM.

Let's make sure that the full photogrammetric image processing always based on the SPM. The term "full photogrammetric processing" (FPP) means here that the analysis starts from the digital image data and provides the 3D-representation of the observed scene. Any FPP is a *modular sequential process* that includes, at least, image registration, orientation parameter

estimation, corresponding matching, depth map forming, scene recognition and final result outlining. Some image processing techniques are often of use in the FFP, such as image filtering and restoration, edge detection, interpolation and so on. The each stage of this process is executed after the previous stage is done. Thus, the FFP satisfies the conditions (a) and (b). There are some different possible ways to realize the each of the mentioned stages. For instance, there are many different techniques to capture stereo correspondence. We can obtain different FFPs by selecting different alternatives for each of processing modules. Moreover, some of these modules are complex processes with sequential stages and have some different alternatives for the each step. Additionally, some of subprocesses participate in different FFP modules. For example, the edge detection can be applied both to the intensity image at the correspondence matching step and to the depth map at the 3D-scene analysis step. So, we can say that the condition (c) is fulfilled too. Thus, we have all of reasons to develop the digital photogrammetric software using the object-oriented data management.

For object-oriented data management design we propose to use the well-known in the Artificial Intelligent (AI) concept, i.e. the **frame approach**. To proceed from the OOP-terms to the AI terms, we have to change our notions: "class" to *"frame"*, "subclass" to *"slot"* and "object" to *"exeframe"* or *"frame-copy"*. Hierarchically, we can consider any slot as a more simple frame. That makes it possible to determine frames recursively. Moreover, the frames of (k-1)-order can be the slots of the k-order frame. In this way, the desired system behavior is represented as a Semantic Frame Net which elements' interconnections describe the rules of processing of the system events connected with these elements (exeframes).

For developing this conception we need to introduce some basic definitions.

DEFINITION 1. The *Processing Frame (PF)* is the frame of type
<frame>:=<name>{<slot1>,<slot2>,<slot3>,<slot4>},
where the slots have the following interpretations:
<slot1>:=<connections>{<list of frames-correspondents>}
<slot2>:=<procedure>{<list of procedure parameters>}
<slot3>:=<input predicate>{<list of messages>}
<slot4>:=<output>{<list of messages>}

DEFINITION 2. The *Principle Processing Frame* (PPF) is the *PF*, which have all types of slots.

With the help of PPF we can represent the basic objects, which determine the processing, i.e. the objects that are corresponded directly to the basic data types and the processing procedures. The definition of the *basic data type* is intuitive and depends on the proper problem area. In the IP and DP we assume the *Image* to be the basic data type.

DEFINITION 3. The *Hinge Processing Frame* (HPF) is the PF, which has no <slot4>, and which list of connections includes only the PFPs corresponded to the basic data types only.

The result of HPF activity doesn't influence to the further processing. This type of frames is used to display the current information such as histograms, profiles of brightness, densitometry information, 3D representation and so on during the processing.

Thus, the processing scheme is characterized by the PF's interconnection and the HFP serve only to inform user about on the various stages of the system activity.

DEFINITION 4. The *Control Frame* is the PS that has no <slot2>.

These frames provide the possibility of the interactive work with the frame net. Also the usage of such frames makes it possible to break the processing at the any step and change the whole further processing scheme or its part.

DEFINITION 5. The A*synchrony Frame Net* is the PF net, which functioning determines by the results of the processing of the <slot3> messages list of each PF.

It is usable to define, at least, two theoretic-set operations: **union** and **intersection**.

If one unites two or more frames then the resultant frame has all of slots were in the initial frames with their initial parameters (except of slots, which are identical by names). For the slots with the same names it will be only one common corresponding slot in the resultant frame. The parameter of such slot is the union of the parameters of the corresponding slots of the initial frames. The name to the resultant frame is given by the system according to the some set of rules (which is the proper for the each certain system) or (in the general case) makes up from the names of initial frames connected by the "OR" symbol. For instance, the well-known image processing scheme - the *background normalization* - unites two usual filtering schemes (with different size of apertures) and the deduction operation.

According to the intersection there are only those slots in the resultant frame, which have been presented in the all of initial frames. Concerning the parameters of these slots there are two types of the frame intersection in the algebra of frames. In the first case, the slots in the resultant frame maintain only those values that were equal in initial frames. If there is no special name appropriation procedure for this frames the resultant name determines by the appropriation of the name that includes the names of initial frames joined by the "AND" symbol. In the second case, the slots in the result frame have the values corresponded to the union of the initial frames' values. If the system has no special name appropriation procedure then the resultant name determines from the initial names with the help of statement "similar with".

It is important that the frame paradigm described above logically encloses our complete object oriented approach. Indeed, from the one side, the principal processing frames of the frame net must be the copies of the classes that represent data types and algorithms in the object-oriented manner. From the other side, it is naturally and easily to implement the proper frames as the graphic interface elements (windows, icons, etc.) and thus represent the whole frame application in accordance with the multidocument interface software standard (for example, Windows MDI).

5. CONCLUSION

In this paper we apply the *Complete Object-Oriented Processing* principle for the photogrammetry-oriented system development. The object-oriented data representation and management are discussed.

The main results are:

- the object-oriented data representation provides the easy simulation of the large data and also data unification and the fractal properties representation for the photogrammetric image processing tasks.
- the any full photogrammetric processing cycle realizes a structured processing model and thus required data management tools.
- the flexible and high automated data management is achieved by using of the semantic frame net.
- the proposed management scheme encloses the COOP means and allows to simplify the visual interface implementation.

This paper states the principles of the photogrammetric object-oriented software design. In future we are going to present our COOP system which will realize the full photogrammetric processing cycle.

6. REFERENCES

[1] **Myron Flickner, Mark Lavin, Sujata Das** *"An object-oriented language for image and vision execution (OLIVE)"*, **1991**, SPIE, Vol. 1568.

[2] **Stepanov A.A., Vizilter Yu.V., Morzeev Yu.V., Zheltov S.Yu.,** *"Complete Object-Oriented Image Processing"*, **1995**, (to be published).

[3] **Skryabin S.V., Zheltov S.Yu., Vizilter Yu.V.,** *"Photogrammetric processing of the large images on PC"*, **1995**, (to be published).