Implementation of Computer Vision Algorithms in DirectShow Technology

Piotr Szczypinski[1], Pawel Pelczynski[2], Dominik Szajerman[3], and Pawel Strumillo[4]

[1] Institute of Electronics, Technical University of Lodz Wolczanska St. 211/215 piotr.szczypinski@p.lodz.pl
[2] Institute of Electronics, Technical University of Lodz Wolczanska St. 211/215 pawel.pelczynski@p.lodz.pl
[3] Institute of Information Technology, Technical University of Lodz Wolczanska St. 211/215 drs@ics.p.lodz.pl
[4] Institute of Electronics, Technical University of Lodz Wolczanska St. 211/215 pawel.strumillo@p.lodz.pl

Summary. In this paper an application of DirectShow software technology in a computer vision system is described. DirectShow (DS) is well suited for image processing and analysis tasks. In the reported study it was successfully applied in a computer stereovision system. Physical cameras of the system are represented by DS source filters connected to image analysis procedures. Original filter prototypes were designed for stereo disparity estimation and scene analysis tasks. Image analysis procedures for scene depth estimation were build and tested. The developed system has proved usefulness of the DirectShow technology in computer vision applications.

1 Introduction

Development of vision systems that employ multiple hardware and software resources is not a straightforward task. Such systems should be of modular structure and have flexible architecture. The modular design is important in projects including a group of programmers working on different aspects of the project. The system should be flexible not to depend on specific hardware or hardware versions and it should automatically adapt to various configurations. Another requirement in vision systems is a need for conversion of data forms and formats. One of such problems is object tracking in image sequences. The input is a video data whilst the output is usually a motion defined by coordinates, motion vectors and rotation angles. Moreover, multicamera vision systems such as stereovision systems require data synchronization and parallel data processing (multithreading, pipelining and graphics processing units general-purpose computing). The requirements of vision systems programming and achieving acceptable programming efficiency in such vision systems

can be satisfied by employing multimedia frameworks - technologies for multimedia software development. The goal of this publication is to present the application of DirectShow multimedia framework in a stereovision system for detection of objects, object localization, tracking and its parametric description.

DirectShow (DS) is a software technology for multimedia applications in Microsoft Windows operating systems [1], [2]. It serves as a framework for development of video and audio processing programs and modules. DS is one of the three multimedia technologies available for Windows platforms. The two other are Video for Windows and Media Foundation [3]. One of the main features of DS is its modularity. DS modules are called filters. They could be developed by different groups of programmers or software producers independently. Filters are dynamically connected into the so called graph in the time of program execution. There is no need to compile all the modules into a single application. An example of DS based video player is shown in Fig.1. It consists of the following filters: video file reader, audio video splitter, decompression of audio and video, and presentation filters.

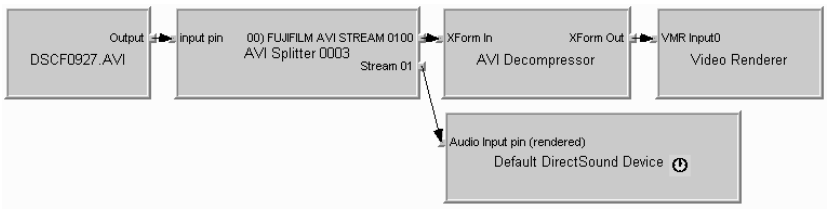

Fig. 1: Graph of filters for video file playback

Filters are implemented as DLL libraries. They are developed according to the specification of Component Object Model (COM) technology. The main assumptions are:

- each filter is a separate COM object,
- each filter is identified by its unique GUID number,
- filter properties are stored in the Windows registry file.

Connection of filters can be a fully automatic process. The type of filter used and its possible connections depend on a set of filters available in a given computer. Thus, the same multimedia file can be played by using different filters or cannot be if the important filter lacks. DS application interacts with filters via intermediary COM object called filter graph manager. It builds a graph by using IGraphBuilder. Graph is a set of connected filters. If necessary, additional transform filters are added to provide compatibility of different data stream formats between filters. The main application can also point out specific filters to be used or even specify connection order. Intermediary object

is also used to control filter functions, e.g. play, rewind or pause of the video stream.

2 Filter programming

DS filter is largely a self-processing module with a significantly greater number of functions and tasks than would result from the operation of the algorithm for processing multimedia data. This module is equipped with interfaces to communicate with other filters, intermediate objects (GraphBuilder) and the main program. These methods are public, available (can be called from outside the filter) and abstract (undefined). Filter interfaces are identified by individual, unique GUID numbers and they have to be compatible with COM technology. A typical interface is the interface of filter terminal, called IPin. Its methods are listed below.

Table 1: An example of IPin interface methods

Method	Description
Connect	Connects the pin to another pin
ReceiveConnection	Accepts a connection from another pin
Disconnect	Breaks the current pin connection
ConnectedTo	Retrieves the pin connected to this pin
ConnectionMediaType	Retrieves the media type for the current pin connection
QueryPinInfo	Retrieves information about the pin
QueryId	Retrieves the pin identifier
QueryAccept	Determines whether the pin accepts a specified media type
EnumMediaTypes	Enumerates the pin's preferred media types
QueryInternalConnections	Retrieves the pins that are connected internally to this pin
EndOfStream	Notifies the pin that no additional data is expected
BeginFlush	Begins a flush operation
EndFlush	Ends a flush operation
NewSegment	Notifies the pin that media samples are grouped as a segment
QueryDirection	Retrieves the direction of the pin (input or output)

IPin interface provides the methods:
- for negotiating data formats (EnumMediaTypes),
- for connecting and disconnecting terminals (Connect, Disconnect),
- for identification (QueryID),
- for data flow control (EndOfStream, BeginFlush, EndFlush) and others.

IPin interface inherits input and output object classes, e.g. CBaseInputPin and CBaseOutputPin. These classes provide some additional methods for multimedia data transfer. A fully functional filter provides multiple interfaces

of the methods. These are the interfaces of individual pins, the interface indicating filter resources, a filter work control interface, the interface of dialog box for filter properties, etc.

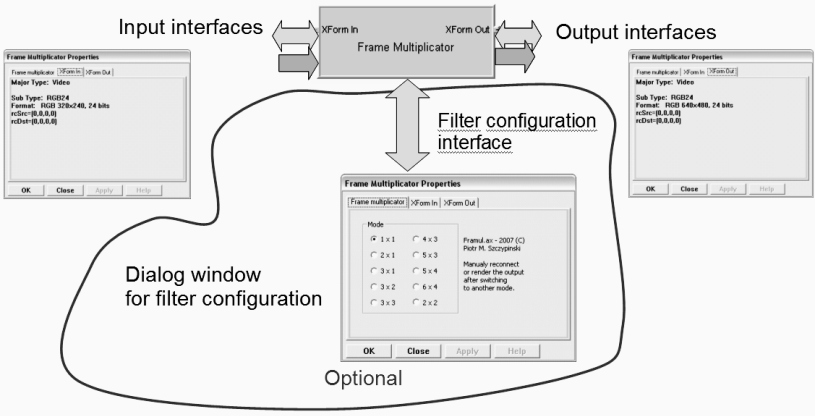

Fig. 2: Filter interfaces

The methods included in the interface must be implemented (written) by the developer creating a filter. It is necessary to provide adequate communication between the main program, an intermediary object and the neighbouring filters. This communication deals with several aspects: filter interconnection strategy, negotiation of transmitted data formats, interface and pin identification, control of data stream playback, processing and synchronization, management of threads for filter control and data transfer, etc. The flexibility of the DS brings with it a significant complication of the process of creating a filter.

Writing the filter from scratch would be extremely laborious without base classes, which were made available by Microsoft. Base classes implement a part of more trivial methods, necessary for the proper functioning of the filters with specific functionality. While creating own filter one creates a new class that inherits from the selected base class. For example, writing a data-processing filter the following base classes can be used: CTransformFilter, CTransformInputPin and CTransformOutputPin. Creating a source filter (eg, providing data to graph from the outside) or output filter (presentation), we can use base classes: CSource and CRenderedInputPin.

Necessary Tools for creating DS filters are: the development environment (it is sufficient to use Visual C++ Express Edition [4]), Windows Platform SDK (includes base classes and tools for testing filters), the program Guidgen, the regsvr32 tool and GraphEdit program (included in the Windows Platform SDK) or Graph Studio. Visual C++ environment is used to create filter's source code and its compilation. The environment allows also to com-

pile a library of base classes available in the Windows Platform SDK. Guidgen program is used to generate GUID numbers identifying the filter and its interfaces. Regsvr32 program is available in Windows system directory and is used to register a new filter in an operating system. Programs GraphEdit and GraphStudio are used for graph visualization, manual filter connection into graphs and testing their performance.

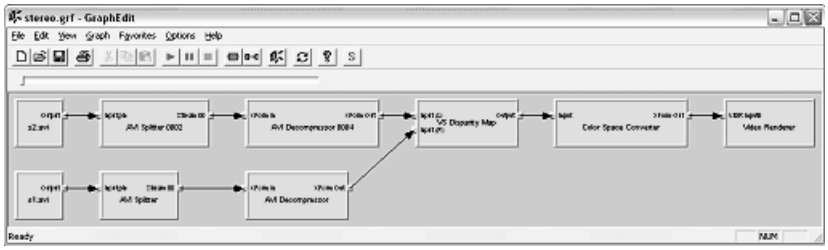

Fig. 3: Filter graph for disparity estimation in GraphEdit window

3 Filter pattern design

The aim of this study was to develop a methodology for the design of vision systems for scene analysis using DirectShow technology. Implementing the system using DS required to develop filters with specific functionality, number of inputs and outputs, control, types of data formats for input and output. The system includes filters for calculating disparity map, object detection and presentation of analysis outcomes. In order to integrate algorithms that were developed earlier, filters with the required functionality need to be defined in DS system. Filter source code was written in C++, using the COM and DS specification, and DS base classes. The written source code is a package of algorithms for image processing and analysis. It is fully functional in terms of DirectShow technology, but the code need to be supplemented in order to obtain the system functionality. Code fragments, where additional commands need to be entered, were marked in the source code.

All data in the DS system are transferred by objects belonging to the class inheriting from the IMediaSample interface. This interface allows to download information about the index to a block of data transmitted and the size of the block (methods: GetPointer and GetActualDataLength). Format of the received data or data sent through the filter is determined in the negotiation process. To send data about data formats CMediaType class objects are used (containing AM_MEDIA_TYPE structure) in the described project. Structure AM_MEDIA_TYPE carries information about the main type (majortype), subtype, and optionally additional information tables set out by the field

Formattype. These additional tables may be e.g. VIDEOINFOHEADER for imaging data.

Developed filters operate on two types of multimedia data, namely: uncompressed image data and a table describing a result of image sequence analysis, such as the scene model parameters.

1. Imaging data: majortype: MEDIATYPE_Video, Subtype: MEDIASUBTYPE_RGB32 or MEDIASUBTYPE_RGB24. These are colour images of 32 or 24 bits per pixel. Note that DS does not define the greyscale image format. 2. The data describing the objects' parameters: majortype: MEDIATYPE_Video, Subtype: MEDIASUBTYPE_NAVIOBJ.

Filter development procedure was defined as a set of the following modifications:

1. Find the header file, which defines the filter GUID identification numbers,
2. Generate a new GUID numbers and replace them in the header file,
3. Find the structures AMOVIESETUP _FILTER and CFactoryTemplate in the source file of the filter, and replace the names of the filters and titles of the properties dialog boxes.
4. Find a piece of code marked "@@@@ Complete code here ... " and write a code of the required calculation algorithm.

4 DS based application for scene depth estimation

On the basis of the prepared filter pattern new filter was designed. It performs initial image preprocessing and disparity estimation in a pair of stereovision images. Image processing operations were implemented in a graphics processing unit (GPU), which is very efficient in vector operations. More details about applying GPU for disparity estimation are given in [5].

The developed DirectShow filter is a dynamic-link library which offers a set of the following image processing functions. They use OpenGL in order to allow setting of the parameters, passing of the input images to the GPU, control of the processing on the GPU and fetching the results from it. The filter has two input pins and one output pin (Fig.4). The input pins are used to retrieve images from the left and right camera. The images can have any resolution and should be in RGB colour format (24 bits per pixel). The output pin transmits the result as the image of the same resolution as the input ones. The output image is also three-channel. The output disparity is stored in the first channel. The processing in the filter consists of a few passes. Every pass is a fragment shader processing of the input textures which contain input data. The result of each pass is stored in the output texture, which is then delivered as an input to the next pass. The processing passes are shown in Fig.5:

1. both input images are converted into greyscale and their geometry is corrected and rectified, the result is stored as a single texture ($I^Y_{l,r}$).

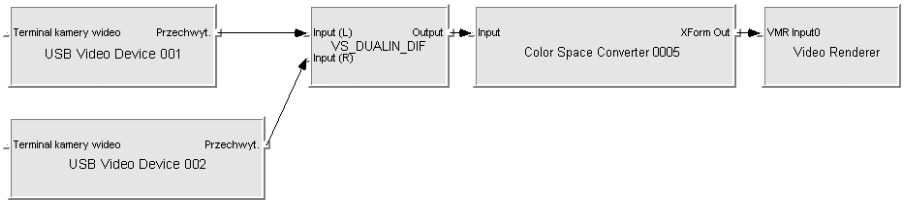

Fig. 4: Filter graph for depth estimation

2. the average brightness is calculated for vertical blocks (size $1 \times b$, where b is the processing parameter - the block size).
3. the average brightness from previous pass is used to calculate the average brightness for blocks of size $b \times b$, next it is subtracted from the original brightness.
4. absolute differences between left and right image are calculated, their count depends on a processing parameter.
5. the sum of the absolute differences in the blocks (size $1 \times b$) is calculated.
6. the SAD in the blocks (size $b \times b$) is calculated.
7. the minimal from the SADs is chosen, its index is stored as the result - calculated disparity value.

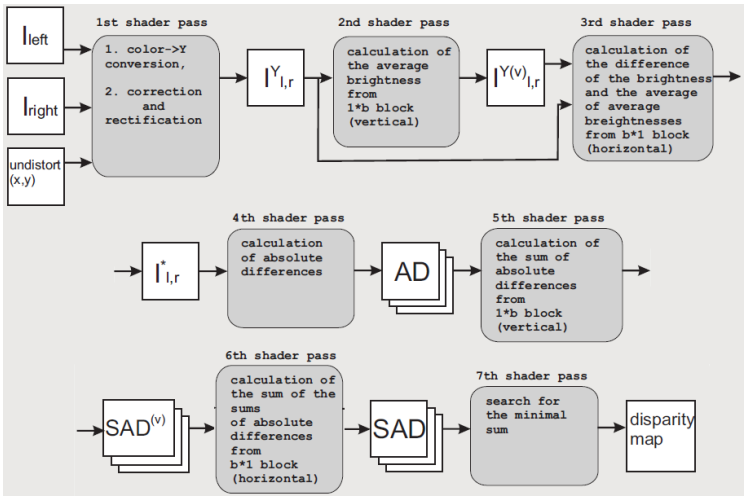

Fig. 5: Data flow in GPU processing

The results of the described image processing procedure are shown in Fig.6. Image disparity map is shown as a grayscale image in which closer scene elements are coded by brighter map regions. GPU implementation makes ex-

ecution of this procedure fast enough to be run in real time. Despite the use of GPU the developed filter is quite universal. It can process images in NVIDIA GPUs series 8000 and higher.

(a) (b) (c)

Fig. 6: Original image a) undistorted image b) and disparity map c)

5 Conclusions

DirectShow software technology was used to implement computer stereo vision system. Stereo cameras were represented as DS source filters connected to the filter for image distortion correction and disparity estimation. Other filter prototypes were designed for further scene analysis procedures. A useful system for scene depth estimation was build and tested. The main problems that were solved while implementing DS multimedia framework were: synchronization of video data obtained from two cameras, definition of the original object describing data format to be transferred between modules and the implementation of advanced image processing filter in GPU platform. It was shown that advanced image processing tasks can be mapped onto DirectShow technology that offers more efficient software development and testing in a group of cooperating programmers.

References

1. Microsoft (2007) MSDN DirectShow documentation. http://msdn.microsoft.com/en-us/library/ms783323.aspx. July 7, 2007
2. Pesce, Mark D (2003) Programming Microsoft DirectShow for Digital Video and Television. Microsoft Press. April 16, 2003
3. http://en.wikipedia.org/wiki/Multimedia_framework
4. http://www.microsoft.com/express
5. Strumio P, Szajerman D, Peczyski P, Materka A (2009) Implementation of Stereo Matching Algorithms on Graphics Processing Units, Image Processing & Communications Challenges (R.S. Chora, A. Zabudowski, Eds.), Academy Publishing House EXIT, Warsaw 2009, pp 286-293.