FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 20, no. 1, April 2007, 93-105

C# Solutions for a Face Detection and Recognition System

Cătălin-Daniel Căleanu and Corina Botoca

Abstract: Key issues on using a new programming language - C# - in implementation of a face detection and recognition (FDR) system are presented. Mainly the following aspects are detailed: how to acquire an image, broadcast a video stream, manipulate a database, and finally, the detection/recognition phase, all in relation with theirs possible C#/.NET solutions. Emphasis was placed on artificial neural network (ANN) methods for face detection/recognition along with C# object oriented implementation proposal.

Keywords: C#/.NET, face detection and recognition.

1 Introduction

In June 2000, Microsoft announced both the .NET platform and a new programming language called C# [1–3]. .NET is a framework that covers all the layers of software development from the operating system up. It actually wraps the operating system, insulating software developed with .NET from most operating system specifics such as file handling and memory allocation. It provides a new application programming interface (API) to the services and APIs of classic Windows operating systems while bringing together a number of disparate technologies that emerged from Microsoft during the late 1990s. It provides the richest level of integration among presentation technologies, component technologies, and data technologies ever seen on a Microsoft platform. This includes COM+ component services, a commitment to XML and object-oriented design, support for new web services protocols such as SOAP, WSDL, and UDDI, etc. .NET framework components are depicted in Figure 1.

Manuscript received September 4, 2006, revised November 25, 2006.

The authors are with Electrical and Electroenergetic Faculty, “Politehnica”’ University Timişoara, V. Parvan no. 2, Timişoara 1900, Romania (e-mails: `[catalin.caleanu, corina.botoca]@etc.utt.ro`).

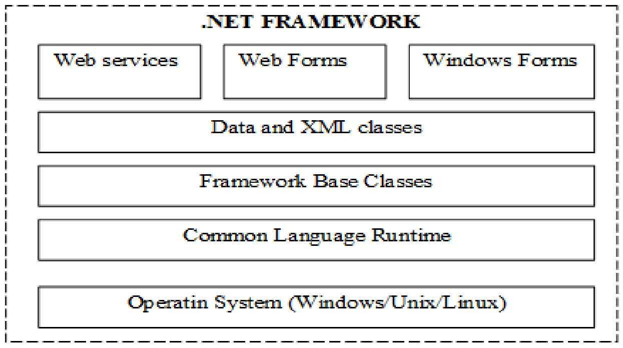

Fig. 1. .NET Framework into its system architectural components.

The programming language of choice for .NET platform is C#. But C# is also an attractive language per se, aside from .NET. The language is powerful, productive, type safe, has a rich and clear syntax and, most importantly, provides a conceptually appealing implementation of the object-oriented paradigm. It is designed to give the optimum blend of simplicity, expressiveness, and performance, pushing beyond the limitations of Java, C and C++ (Figure 2).

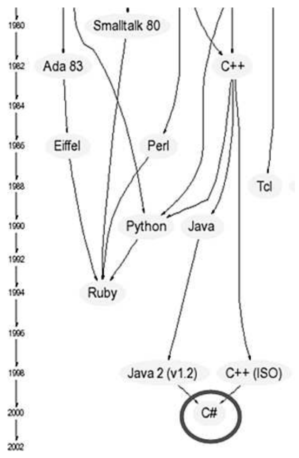

Fig. 2. C# inherits, in principal, from Java and C++.

It inherits from:

- C, the high performance;
- C++, the object-oriented structure;
- Java, the garbage collection and high security;
- Visual Basic, the rapid development.

See [4] for detailed comparison between C# and other programming languages. Having in view the above mentioned characteristics, C# and the .NET might represent an attractive alternative for a facial detection and recognition (FDR) system implementation.

2 Requirements for a FDR System

Usually, a facial detection and recognition (FDR) system has the following capabilities [5]:

a) Interfacing with a video source for grabbing facial images. This implies the possibilities of starting/stopping a video stream and capturing some frames within it;
b) Automatic detection or manual selection of human faces may be found within the scene;
c) Manipulate (create, add, delete) a database of faces. It is also recommended the possibility of visua] database browsing;
d) Launching the recognition process by comparing the face previously detected with the database's faces. The classification operation is usually preceded by some image preprocessing operations and features extraction.

Figure 3 summarizes basic operations required by a facial detection and recognition system [6].

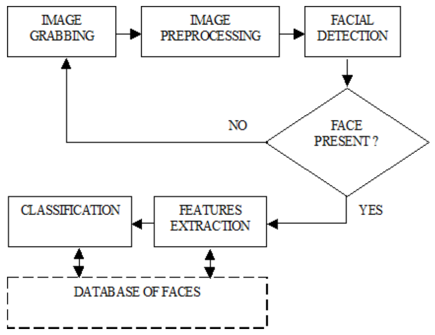

Fig. 3. Operations required by a facial detection and recognition system.

In the following, FDR's blocks set possible C# implementation are detailed.

3 Image Acquisition and Web Broadcasting

Image acquisition includes the first two stages depicted in Figure 3, namely image grabbing and preprocessing. Eventually, at this step, the video stream could be broadcasted, in real time, over Internet. Considering the coding difficulties, using C# together with a videocapture device (videocamera, webcam), grabbing a still image from the video stream and, eventually, Internet broadcast it via a web server, is nowdays the most problematic aspect of the C# FDR implementation. Most of the problems are promised to be solved in the near future by DirectX 10 / Windows Longhorn launching. It seems that there are only two ways to communicate with a videocapture device:

a) Use DirectX's component called DirectShow. DirectShow is a standardized Microsoft Win32 API to use any compliant movie or video device from your application. Unfortunately, DirectX doesn't provide yet DirectShow component for C#. To overcome this problem one has to take into account that DirectShow is exposed as COM components and interfaces:
   - DirectShow custom interfaces - mainly for C++ programmers. There is no direct way to access the custom DirectShow interfaces. It has to be using Interop with one of these approaches: use managed Extensions for C++ or rewrites all the interfaces from to C#. Although most DirectShow interfaces are not very complex this has some drawbacks, e.g. the amount of initial work for rewriting the interfaces and the difficulty of Interop understanding and correct using.
   - DirectShow VB components - designed for VB6, provides a type library.

b) Use a 3rd party cam server. In this case the project could be based on the Webcam COM component developed in the C++ ATL Web Service. DOT NET COM Interop must be used to import it in the C# project. Compared to the C++ ATL Web Service, we need only three lines of code to get the picture from the COM component and return it to the client.

More on this topic could be found in [7].

4 Face Detection and Classification

4.1 The face detection problem

Considering an image representing a frame taken from a video stream or a graphic file selected from a database; the problem of face detection consist of finding the spatial location within the scene where human faces are located. This problem is quite challenging due numerous issues, e.g. pose, presence or absence of structural components, facial expression, occlusion, orientation, imaging conditions. According to [8], methods employed by face detection could be roughly classified in:

- KNOWLEDGE-BASED METHODS.
  Usually rule-based methods, using multiresolution, these methods encode human knowledge of what constitutes a typical by capturing the relationships between facial features. Such an approach is presented in [9].
- FEATURE INVARIANT METHODS.
  Include facial features, texture, skin colour, in order to find structural features. See [10] as an example of these algorithms.
- TEMPLATE MATCHING METHODS.
  Algorithms like predefined or deformable face template compute the correlations between an input image and the stored patterns. More details could be found in [11].
- APPEARANCE-BASED METHODS.
  Employs eigenfaces [12], neural networks [13], support vector machine [14] or hidden markov models [15]. Here, the models are learned from a set of training images which should capture the representative variability of facial appearance.

4.2 The face recognition problem

Many paradigms are available for implementing the recognition/classification phase. Some of the most important are briefly discussed in the following.

- GEOMETRIC FEATURE BASED MATCHING.
  Brunelli and Poggio in 1992 extended Kanade's algorithm and used "Geometric Feature based Matching" for face recognition [16], [17]. The basic idea behind their algorithm was to describe the overall configuration of the face by a vector of numerical data representing the relative position and size of the main facial features: eyes and eyebrows, nose and mouth.
- EIGENFACES.
  Eigenfaces proposed by Turk et al. [12] are a set of orthonormal basis vectors computed from a collection of training face images. They provide a basis of low dimensional representation of the facial images and are optimal in the minimum least square error sense.
- SUPPORT VECTOR MACHINES.
  In 2001, Guo et al. [18], incorporated Support Vector Machines (SVM's) with binary tree recognition for multi-class recognition. More on this topic in [19].
- MATCHING INEXACT GRAPHS.
  In 2001 Cesar et al. [20] approached facial feature recognition as a problem of matching inexact graphs where the graphs were built from regions and relationships between regions in an image.

- DEPTH AND TEXTURE MAPS.
  Texture coding provides information about facial regions with little geometric structure like hair, forehead and eyebrows whereas a depth map provides us with information about regions with little texture such as chin, jaw line and cheeks. Considering this fact, BenAbdelkader et al. proposed that the accuracy of FRT systems can be improved by considering not only the texture map but also the depth map [21].
- MULTIRESOLUTION ANALYSIS.
  Ekenel and Sankur proposed multiresolution facial recognition in [22]. They employ multiresolution analysis to decompose the image into its subbands prior to the subspace operations such as principal or independent component analysis.
- GABOR FEATURE CLASSIFIER.
  Liu et al. [23] describe a novel Gabor Feature Classifier (GFC) method for face recognition. The kernels of Gabor wavelets are similar to the 2D receptive field profiles of the mammalian cortical simple cells and exhibit desirable characteristics of spatial locality and orientation selectivity.

4.3 C# solution for face detection and recognition

As it results from facial detection and recognition literature survey [8]- [23], one of the most promising approaches for both detection and recognition phase is the artificial neural network (ANN) paradigm. Various ANN architectures were employed for the above mentioned task. Among them:

- Multilayer perceptron approach [24];
- Convolutional Neural-Network Approach [25];
- Probabilistic Decision-Based Neural Networks [26];
- Radial Basis Function Neural Networks [27];
- Fuzzy ART Neural Networks [28].

Not only the promising results obtained by the ANN approach but also the easy object-oriented implementation makes neural networks the best candidate for a C# implementation. The object-oriented (OO) representation for neural networks endows them with a flexibility which allows various architectures to be defined and various algorithms to be assigned to those architectures.

Many attempts were made in the neural networks object-oriented implementation. For example in [29] a neural network is described in terms of such concepts of object-oriented concurrent languages as objects, instantiation, inheritance, message-passing, and concurrency. In [30] are comparatively presented two simulation software packages, OpenSimulator and Sesame for extensible and modular

ANN implementation. Neural Network Objects (NNO) is a C++ library specialized on selforganizing incremental networks [31]. MLC++ library [32] is designed through sets of independent units, encapsulating in different classes different concepts related to learning machines. NEURObjects [33] is a library classes for neural network development with the main goal in supporting experimental research in neural networks and fast prototyping of inductive machine learning applications.

All the above solutions were developed in C++. Java is also a good OO choice and a lot of implementations already exists.

As we previously pointed, C# combines the strengths from C, C++ and Java. Therefore a C# ANN implementation is expected to be highly productive and reliable. To our knowledge only few tryings were made toward a C# ANN implementation. Among them C# Neural Network package [34] and Neuro.NET v.2.0 [35].

Unfortunately all previous mentioned ANN implementations suffer some drawbacks: some need remarkable computational resources, as well as considerable training time for the software developer. Most of them were developed with emphasis on specific neural network models, reducing thus the code generality.

In our view, the best class-based ANN implementation is given by Neural Network Toolbox [36] from MathWorks MATLAB [37]. The strongest point of this ANN software implementation is the definition of the network class, sufficiently general to create approximately 15 different types of artificial neural networks, feedforward and feedback, supervised and unsupervised architectures. Feedforward Backpropagation, Elman, Hopfield, Radial Basis Self-Organizing Map are only few examples of ANN architectures available in MATLAB.

Comparison to a similar C# implementation yields some advantages: faster execution speed, less computer resources and a non-proprietary scripting language (MATLAB is quite expensive!).

Having in view the previous work on OO ANN implementation, in part discussed above, we propose, in Figure 4, a C# class hierarchy sufficiently general to implement any kind of ANN architecture.

5 Database Implementation

The FDR's Database component is required for storing and retrieving human facial images along with additional information (Figure 5) on computer's local folders, via a computer network or even trough Internet.

A database is similar to a data file in that it is a storage place for data. Like a data file, a database does not present information directly to a user; the user runs an application that accesses data from the database and presents it to the user in an understandable format.

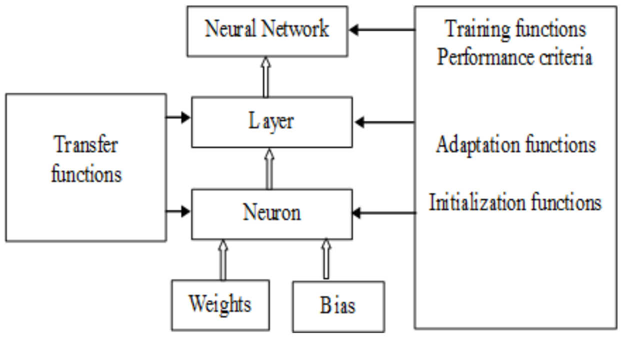

Fig. 4. A class hierarchy for a C# ANN implementation.

Although the FDR database accessing mechanism could be implemented using separate folders and files for each available image of a person, using a dedicated server solution and a database API yields a more powerful and secure solution having multiple advantages [38]:

- In database systems data is more highly organized;
- Robust data exchange among heterogeneous computing environments;
- Native support for the transmission of data sets across firewalls using HTTP;
- There are no duplicate pieces of data;
- Requirement to rapidly scale while supporting a centralized data store;
- Related pieces of data are grouped together in a single structure or record, and relationships can be defined between these structures and records.

Besides, when working with data files, an application must be coded to work with the specific structure of each data file.

In contrast, a database contains a catalogue that applications use to determine how data is organized. A database could be accessed using Structured Query Language (SQL), which is a standard language supported by most database software including SQL Server, Access, and Oracle. Microsoft offers a wide variety of solutions for database servers ranging from classical Microsoft SQL Server 2000 to the newest member Microsoft SQL Server 2005 - Community Technology Preview - Enterprise Edition launched in June 2005. Also free limited versions are shipped with the .NET SDK/Visual Studio .NET. They come with a stand-alone desktop database server known as the Microsoft SQL Server Desktop Engine (MSDE).

When using C#/.NET as implementation language/framework for the FDR system an obviously solution for database API is represented by ADO.NET. It is an elegant, easy-to-use database API for managed applications. ADO.NET is exposed as a set of classes in the .NET Framework class library's System.Data namespace and its descendants. The stated goals of ADO.NET are to [39]:

- Provide a disconnected (offline) data architecture in addition to supporting connected operation;
- Integrate tightly with XML;
- Interact with a variety of data sources through a common data representation;

As an integral part of the .NET framework, it shares many of its features such as multi-language support, garbage collection, just-in-time compilation, object-oriented design, and dynamic caching, etc. . The database component could be implemented, at least, in two ways:

- One of them is that you can save the pictures in a folder and store the path to each one in a database or file;
- The other one is to store the entire file into a database, along with its file name.

Each of them has its ups and downs:

- If you save your files to a folder, you might accidentally delete a file from that folder. If this happens, you will end up with a broken link in your database or configuration file;
- If you store your files into a database, you can enforce security by using the security settings of the database. Also, there are no broken links ever. However, the database storage space is more expensive.

To access and manipulate data from the data store, you'll work through an existing data provider. The .NET data providers link the data store and your application. The .NET Framework includes two data providers for your use, depending on which data store you'll be accessing, as follows:

- OLE DB .NET Data ProviderUsed to access any OLE DB-compliant data store;
- SQL Server .NET Data ProviderUsed to access Microsoft SQL Server 7 or later data stores.

Each of the data providers holds an implementation of the following classes, which form the core of the provider: *Connection*- used to establish the connection to the data store, *Command*- used to execute commands on the data store, *DataReader* -used to access data in a forward-only, read-only form, *DataAdapter* -used to access data in a read/write form and to manage updates of data.

For practical implementation of FDR SQL Server database from a Visual Studio .NET Windows application, just few steps are required:

- Create a connection to the server;

- Create a data adapter that includes a command used to access the data from a database object such as a table or a view;
- Create a DataSet object that would serve as the intermediary between the data in the database and the controls of a graphical application;
- Fill the DataSet object with the data adapter;
- Bind Windows control(s) to the DataSet object.

See more in [40–43] regarding code details on the above topic.

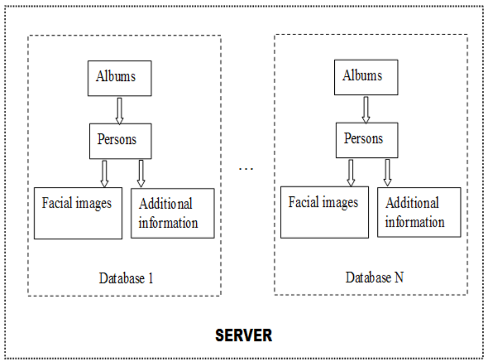

Fig. 5. A possible structure for a facial image database stored on a database server.

6 Conclusion

Solutions towards a C#/.NET implementation for a real-time neural network-based face detection and recognition system are presented here. First, FDR's key requirements were identified. Then, several aspects have been addressed explicitly: C# videocapture device interfacing and web broadcasting, an overview of face detection and recognition methods, especially those which make use of neural networks, a possible C# neural network implementation and how to develop an ADO.NET facial database.

Finally, a face recognition system was developed and tested in MATLAB and then implemented in C# (Figure 6).

It consists of an Interest Operator feature extraction stage followed by a neural network (MLP type) classifier. The experimental results, obtained using the well known ORL public database of faces, are presented in Table 1. The terms e_t and t_p denotes the classification error over the test data set respectively the total processing time (2.66 GHz Intel P4 CPU).

Almost all ORL results are reported in literature using 200 training images and 200 test images (ORL 50/50) randomly selected. In spite of this fact, it is

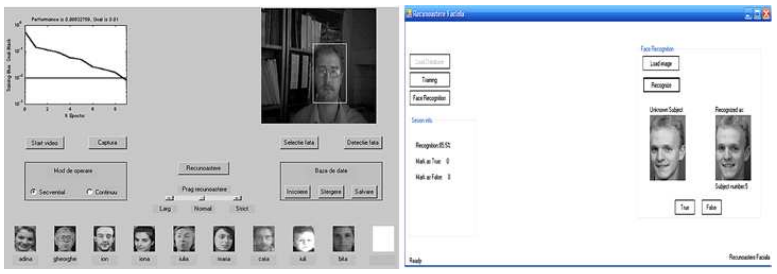

Fig. 6. The FDR MATLAB and C# implementation

Table 1. Test error for ORL 50/50 database

Implementation	Test errors over 5 experiments [%]					min.	max.	std. dev.	e_t [%]	t_p [s]
MATLAB	4.5	5.5	5.5	9.5	4.5	4.5	9.5	2.1	5.9	42
C Sharp	7.5	4.5	6.5	10	5.0	4.5	10	2.2	6.7	19

still difficult to make a fair comparison with other approaches because of different numbers of runs used to compute the test error rates. For example in [44], a mean error rate of 4% based on 20 runs is reported. In contrast, an error rate of 2.5% is reported in [45] based on ten runs, yet in [46] 0% error rate is claimed but it is not clear how many runs the result is based upon. As it could be remarked from Table 1, the C# implementation is more than two times faster in comparison with MATLAB version while the test error is almost the same, arround 6%. In fact, we were more preoccupied by the C# FDR software implementation issues: simplicity, expressiveness, versatility and performance. No special care has been taken in feature extraction/classficator parameters optimization. Future work in this direction might bring lower test error levels.

As a general conclusion we affirm that the new programming language, C#, and the .NET framework are good candidates for a FDR system implementation.

Acknowledgments

The authors acknowledges the support of CNCSIS Grants AT69/2005 and AT41/2006 for some parts of this research.

References

[1] T. Archer, *Inside C#, 2nd Edition*. Microsoft Press, 2002.
[2] D. S. Platt, *Introducing Microsoft .NET, 2nd Edition.* Microsoft Press, 2002.
[3] L. O'Brien and B. Eckel, *Thinking in C#*. Prentice Hall, 2003.

[4] E. Gunnerson, *A Programmer's Introduction To C#.* Apress, 2001.

[5] C. D. Caleanu, "Facial recognition based on parallel neural processing and interest operator," Ph.D. dissertation, University POLITEHNICA Timisoara, 2001.

[6] ——, "Conventional versus neural networks techniques for facial recognition," in *Proc. of the Symposium of Electronics and Telecommunications, Etc2000*, Timisoara, 2000, pp. 61–64.

[7] The Code Project. [Online]. Available: www.codeproject.com

[8] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *EEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34–58, Jan. 2002.

[9] G. Yang and T. S. Huang, "Human face detection in complex background," *Pattern Recognition*, vol. 27, no. 1, pp. 53–63, 1994.

[10] R. Kjeldsen and J. Kender, "Finding skin in color images," in *Proc. Second Int'l Conf. Automatic Face and Gesture Recognition*, 1996, pp. 312–317.

[11] A. Lanitis, C. Taylor, and T. Cootes, "An automatic face identification system using flexible appearance models," *Image and Vision Computing*, vol. 13, no. 5, pp. 393–401, 1995.

[12] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[13] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, Jan. 1998.

[14] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: An application to face detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997, pp. 130–136.

[15] A. Rajagopalan, K. Kumar, J. Karlekar, R. Manivasakan, M. Patil, U. Desai, P. Poonacha, and S. Chaudhuri, "Finding faces in photographs," in *Proc. Sixth IEEE Int'l Conf. Computer Vision*, 1998, pp. 640–645.

[16] T. Kanade, "Picture processing by computer complex and recognition of human faces," Ph.D. dissertation, Department of Information Science, Kyoto University, 1973.

[17] R. Brunelli and T. Poggio, "Face recognition through geometrical features," in *European Conference on Computer Vision (ECCV)*, 1992, pp. 792–800.

[18] G. Guo, S. Li, and K. Chan, "Support vector machines for face recognition," *Image and Vision Computing*, no. 19, pp. 631–638, 2001.

[19] Y. Lia, S. Gong, J. Sherrah, and H. Liddell, "Support vector machine based multi-view face recognition and detection," *Image and Vision Computing*, no. 22, pp. 413–427, 2004.

[20] R. Cesar, E. Bengoetxea, and I. Bloch, "Inexact graph matching using stochastic optimization techniques for facial feature recognition," in *International Conference on Pattern Recognition (ICPR)*, Quebec, Canada, 2002.

[21] C. BenAbdelkader and P. Griffin, "Comparing and combining depth and texture cues for face recognition," *Image and Vision Computing*, no. 23, pp. 339–352, 2005.

[22] H. K. Ekenel and B. Sankur, "Multiresolution face recognition," *Image and Vision Computing*, no. 23, pp. 1–9, 2005.

[23] C. Li and H. Wechsler, "A gabor feature classifier for face recognition," in *Eighth IEEE International Conference on Computer Vision*, vol. 2, July 7–14, 2001, pp. 270–275.

[24] C. Caleanu, “Facial recognition using committee of neural networks,” in *5th Seminar on Neural Network Applications in Electrical Engineering (NEUREL-2000)*, Belgrade, Yugoslavia, 2000.

[25] I. Lawrence, C. Giles, A. Tsoi, and A. Bock, “Face recognition: A convolutional neural-network approach,” *IEEE Trans. Neural Networks*, vol. 8, pp. 98–113, 1997.

[26] S. Lin, S. Kung, and L. Lin, “Face recognition/detection by probabilistic decision-based neural networks,” *IEEE Trans. Neural Networks*, vol. 8, pp. 113–123, 1997.

[27] K. Sato, S. Shah, and L. Aggarwal, “Partial face recognition using radial basis function network,” in *Proceengs of the third IEEE International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, Apr. 14–16, 1998, pp. 288–292.

[28] L. Pessoa and A. Leitao, “Complex cell prototype representation for face recognition,” *IEEE Trans. Neural Networks*, vol. 10, no. 6, pp. 1528–1531, 1999.

[29] A. Weitzenfeld, “Nsl, neural simulation language, version 2.0,” Center for Neural Engineering, University of Southern California, Tech. Rep. 90-01, 1990.

[30] G. Thimm, R. Grau, and E. Fiesler, “Modular object-oriented neural network simulators and topology generalizations,” in *Proceedings of the International Conference on Artificial Neural Networks, ICANN 94*, M. Marinaro and P. G. Morasso, Eds. Springer-Verlag, May 26–29, 1994, pp. 747–750.

[31] M. Kunze and J. Steffens, “The neural network objects,” in *Proceedings of the fifth AIHENP Workshop*, Lausanne, 1996.

[32] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger, “Mlc++: a machine learning library in c++,” in *Tools with Artificial Intelligence ’94*, IEEE Computer Society, 1994, pp. 740–743.

[33] [Online]. Available: http://www.disi.unige.it/person/ValentiniG/NEURObjects

[34] [Online]. Available: http://franck.fleurey.free.fr/NeuralNetwork/home.htm

[35] [Online]. Available: http://xpidea.com

[36] M. B. Howard Demuth, *Neural Network Toolbox User’s Guide*. The MathWorks, Inc., June 2004.

[37] [Online]. Available: www.mathworks.com

[38] J. Price, *Mastering C# Database Programming*. Sybex, 2003.

[39] J. P. McManus and C. Kinsman, *C# Developer’s Guide to ASP.NET, XML, and ADO.NET*. Addison Wesley, 2002.

[40] B. Joshi and P. D. et al., *Professional ADO.NET Programming*. Wrox, 2002.

[41] R. M. Riordan, *Microsoft ADO .NET Step by Step*. Microsoft Press, 2002.

[42] B. Hamilton and M. MacDonald, *ADO.NET in a Nutshell*. O’Reilly, 2003.

[43] M. Williams, *Microsoft Visual C#.NET (core reference)*. Microsoft Press, 2002.

[44] M. Wang and S. Chen, “Enhanced fmam based on empirical kernel map,” *IEEE Trans. Neural Networks*, vol. 16, no. 3, pp. 557–564, May 2005.

[45] M. E. Er, W. Chen, and S. Wu, “High speed recognition based on discrete cosine transform and rbf neural networks,” *IEEE Trans. Neural Networks*, vol. 16, no. 3, pp. 679–691, May 2005.

[46] H. Zhang, B. Zhang, W. Huang, and Q. Tian, “Gabor wavelet associative memory for face recognition,” *IEEE Trans. Neural Networks*, vol. 16, no. 1, p. 275278, Jan. 2005.