Michał PODPORA

Opole University of Technology

Computer vision in parallel computing

Abstract. *Computer vision is known to be a CPU power consuming task. Real-time video capturing, filtering and converting take so much time that creating a substantial artificial intelligence algorithm seems to be unrealisable. In this article author considers using a cluster in a computer vision project, advantages and disadvantages of this solution. He explains some ideas about minimization of amount of data and other methods of speeding up the algorithm.*

Streszczenie. *Systemy wizyjne bywają bardzo wymagające pod względem mocy obliczeniowej. Przechwytywanie, filtrowanie i konwersja obrazu w czasie rzeczywistym zajmują tyle czasu, że stworzenie bardziej złożonego algorytmu sztucznej inteligencji wydaje się być niewykonalne. W tym artykule autor rozważa użycie klastra komputerowego jako sposobu na rozproszenie algorytmu analizującego obraz pozyskiwany z systemu wizyjnego, omawia zalety i wady tego pomysłu, proponuje kilka pomysłów na minimalizację czasu komunikacji i przyspieszenia działania algorytmu.*

Keywords: parallel computing, computer vision, signal processing, neural networks
Słowa kluczowe: przetwarzanie równoległe, systemy wizyjne, sieci neuronowe

Introduction

Almost every single being on our planet uses sight as one of its most important senses. We already know how the retina works and more or less how the brain acquires and identifies the visual "data", but our knowledge is still insufficient to build a complete real model.

Many scientists and hobbyists have struggled against lack of CPU power for all the real-time video capturing, filtering, converting, processing, etc. not mentioning e.g. pattern recognition algorithms.

Why parallel computing

Nowadays, CPU's are much more efficient, but a single PC still seems to be not enough for such a complex job. That's why the author has chosen a different solution (surprisingly not a popular one in computer vision) – to use a cluster instead of a "single" PC.

Computation time vs. communication time

Writing an application for a cluster is always a great challenge. Moreover, computer vision is based on real-time video stream, whereas one of the main problems of parallel computing is minimization of communication time between nodes of a cluster. This objective can be achieved in many ways. First of all, the computer with the capturing device should be a part of the cluster (as input node) – otherwise, the communication between this computer and the cluster would be a bottleneck of entire process. Another idea is to reduce all time-consuming communication by eliminating video streams and still frames and reduction of amount of data sent between nodes.

Communication – no frames?

Using video stream as input signal does not mean sending and receiving video data between all nodes of a cluster. In fact, video data can be processed on the input node "in real-time", while all other nodes can work asynchronically [2] basing on reduced and selected data from the input node. It means that reactions of application can possibly happen a moment after specific stimulus – like in real world – it is possible to catch a sight of something, and a moment later to "get the visual" (e.g. when travelling by bus).

The solution of this problem is brought by the nature – eye is only a complex receptor: photoreceptor nerve cells in retina convert light to electrical signals sent to the brain via optic nerve. This process should not be compared with a video camera. The border between a video camera and a PC should rather be understood as the border between pupil and retina than between optic nerve and brain. The captured signal should not "fall" directly into the neural network. It should be converted to a form that would be comprehensible for a neural network.

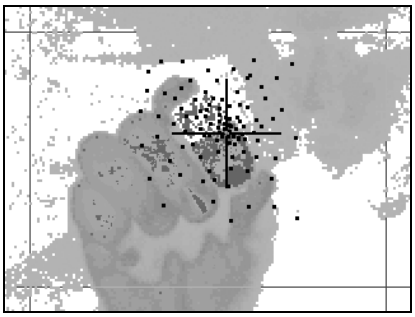

Fig.1. An object I'm holding is being "observed" by the "yellow spot"

Not all details in the field of vision are watched with the same attention at the same time. Retina has a small depression called yellow spot with a rod-free region (fovea centralis) with more closely packed cones providing the sharpest and most detailed information [1]. This sentence is the key to reducing the amount of data. Let's define a "virtual yellow spot" – a point of the most detailed information [5]. It is visualized on (1) as the big black cross. Neural network could modify spots' coordinates to "watch" details of the scene. Inputs of first layers should get their values (of e.g. colour or movement) in specified points in the neighbourhood of the yellow spot – on (1) some of them are shown as black points.

"Yellow spot" – idea for reducing amount of data

All inputs of a specified artificial neural network within a node of the cluster should acquire data from the same source (i.e. "colour matrix", "move matrix", etc.). The algorithm of the application becomes much more simple to understand if the sources are not mixed. It would be hard to determine if e.g. network analysing movement, with one or two inputs passing values from the "colour matrix", would improve or aggravate outcomes. (Example: All human beings walking into the laboratory are supposed to wear white clothes. What if an object is moving like a human but it is not dressed white? Shall it not be recognized as human?)

Virtual yellow spot is an idea for reducing amount of transferred data between the input node and other nodes of

a cluster. It is useful only if the data sources prepared by the input node represent really substantial aspects of observed scene.

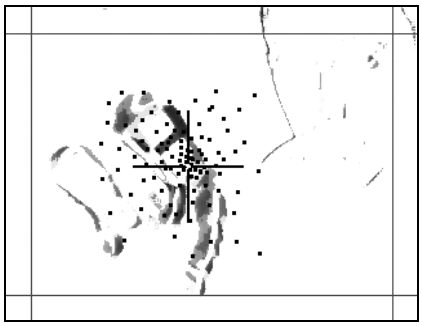

Fig.2. The difference between two subsequential frames ("move matrix")

The basic and most popular approach to the movement analysis is comparison of two subsequential frames [4]. This is the easiest possible way to set the yellow spots' coordinates (2). This solution is suitable if the algorithm is supposed to search for movement or analyse moving objects. If the task is more complicated (like keeping the focus of one object regardless of the movement of other objects in the scene), the algorithm would be far more complicated and other types of data sources ("connected" to other neural networks) should be involved in the election of the new coordinates of the yellow spot.

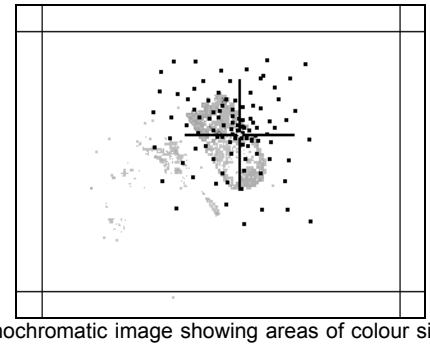

Fig.3. Monochromatic image showing areas of colour similar to the colour of the pixel at yellow spot's coordinates

Colour can be a very helpful data source in object recognition. The amount of data can be reduced by selecting one colour and understanding the image as monochromatic one. Selecting the proper colour is a separate issue – the colour can be chosen to show e.g. contrasting objects or areas of the image with colour similar to the colour of the pixel at yellow spots' coordinates (3).

Neural networks should be able to chose the data source dynamically at startup – it would enable the designer of the application to easily implement new data structures for the neural networks.

Cluster virtual topology and communication model

Designing a parallel algorithm requires a due consideration to the broad lines of the clusters' topology. Defining virtual connections between nodes can appreciably speed up (or slow down if done improperly) whole application. Topology proposed by the author for this particular use is modelled on the biological specificity of "understanding the vision". Beside the input node (which captures and converts video stream to a form acceptable for inputs of a 1st layer of a neural network) there are other nodes in the cluster – getting data from the input node in a particular purpose: to process colour or shape or movement – just like it happens in nerve centres in a brain. These nodes can not only influence coordinates of the "yellow spot" but also generate some output information describing the input.

Cluster virtual topology defines how the nodes communicate with each other – where should they look for input values and where should they send the output. The most intuitive topology seems to be the star topology (4a). The input node is the centre of the cluster and all other nodes sending and receiving data with the input node. This solution corresponds with our notion of human brain – information sent from the photoreceptor to the specified areas in the brain. Unfortunately reality is far more complicated than this simplification. Besides, the more nodes there are in the cluster, the more time input node needs for the communication.

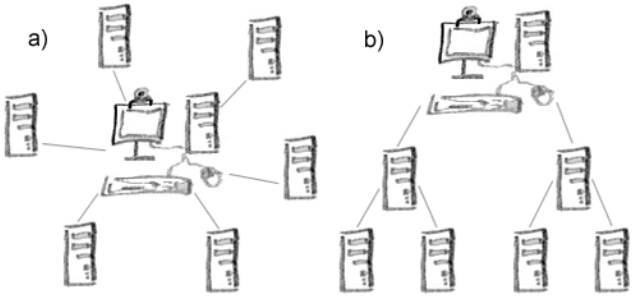

Fig.4. Cluster virtual topology – star (a) and tree (b)

Tree topology (4b) makes the algorithm more scalable – although the input node remains still the centre of the topology, it doesn't have to take care of the whole communication. The problem is that load balancing becomes more complicated issue, also nodes should process only data available on "their branch". However this topology is more adequate to the biological model: the visual data is neither divided into parts for nerve centres, nor sent as a whole to all of them. In fact, it flows through different types of cells and layers from retina to the neocortex. Optic nerve, formed by the fibres of retinal ganglion cells, terminate at six locations in the brain. 90% of those project to a location called lateral geniculate nuclei (LGN). Fibres from LGN cells project to different layers of V1 (striate cortex), and some of V1's layers project to extrastriate cortex [3]. Deeper analysis of this issue helps to interpret the entire process of "understanding the vision" and may be useful to create a practical realisation of an "artificial sight".

Sight, understood as a semi-sequential complex process of parallel analysis and interpretation of visual input data, can be realized by a cluster virtual topology similar to the tree topology, modified to cope with differentiated request for CPU power for various tasks (5).

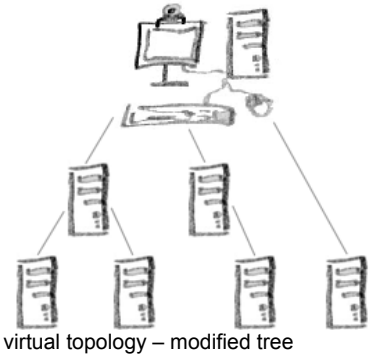

Fig.5. Cluster virtual topology – modified tree

Algorithm – synchronous vs. asynchronous

Synchronous algorithm does have some advantages - it is always clear what and when should occur, when to transmit, when to receive, and it is easier to write one. It has one disadvantage – if one node is ready with its' task (the output is ready to be transmitted), it must wait for the algorithm to allow to send the output.

An asynchronous algorithm is more complex, but it uses a manager which controls communication. If a node is ready, it doesn't have to wait – it asks the manager if it is possible to exchange input and output data and then it can continue processing – no CPU time is wasted. Use of an asynchronous algorithm [2] for the communication speeds up overall performance.

In this particular use, it is not necessary to have the neural networks' output "at once" (i.e. synchronically – before the next video frame occurs). Many programmers have tried such approach – the more nodes the longer it takes to get the output (i.e. to allow processing of next frame before the actual one is done). Processing of the next frame can, however, begin before the output is ready.

Practical implementation

A practical implementation was written to check the overall performance and measure the communication time. The cluster virtual topology was implemented as tree topology (5), an asynchronous algorithm was used for the communication [2] and "virtual yellow spot" [5] was implemented. Although the application is not ready yet, first series of measurements could already be made [6] to verify previously described ideas.

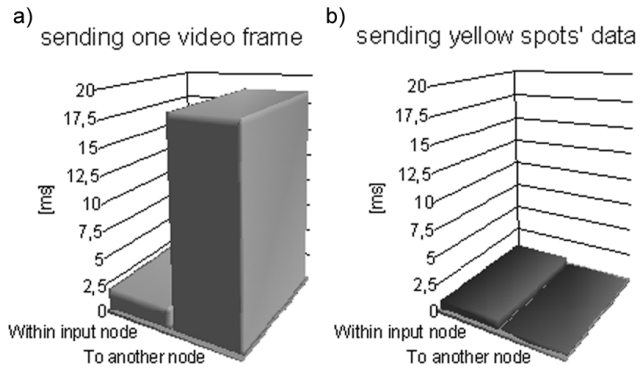

Fig.6. Comparison of communication time

An algorithm processing video frames has been implemented (320x240 pixels, 16bpp, RGB colour images) just for testing the communication time. All analysis and processing was made on video frames, whole frames were transmitted to another node (6a). If the algorithm has been run using only the input node (without the cluster) average communication time for a frame was 2,08 ms (passing one frame only within the input node). If the application used one remote node for processing and/or analysis of video frames, the communication time amounted to 18,60 ms.

If the algorithm used the previously described idea of "yellow spot" (and it was not passing video frames, but selected attributes of selected 200 points of a frame), the communication time lowered from 0,84 ms (transmitting within input node) to 0,16 ms (between nodes). (6b)

Future work and promising directions

At present, the application can capture the input video stream, generate desired simplified structures and send them to other nodes. Current version of algorithm recognises only the simplest possible features of the scene. It can detect where is the most intense movement and basing on this criterion vote for new coordinates of "yellow spot". Also second feature is currently being implemented – detection of the most contrasting colour and (if there is no significant movement at the moment) also vote for new coordinates. When the colour feature is ready, the second neural network would be needed, which means that optimal size of the cluster would be 3 nodes ("input node", movement analysing "node 1", colour analysing "node 2"). After implementing some more features (shape analysis seems to be the most complicated one at the moment), asynchronous communication manager would be useful.

At this stage artificial neural networks will be improved. Best possible learning method will be chosen and implemented.

Summary

Even though scientists and hobbyists seem to avoid parallel computing in computer vision, this union does offer new possibilities.

REFERENCES

[1] Hecht E., Optics, *Addison Wesley*, 4th edition (2002)

[2] Karbowski A., Niewiadomska - Szynkiewicz E., Obliczenia równoległe i rozproszone, *Oficyna Wydawnicza Politechniki Warszawskiej*, (2001)

[3] Mather G., Foundations of Perception, *Psychology Press,* (2006)

[4] Mather G., Introduction to Motion Perception, *available: http://www.lifesci.sussex.ac.uk/home/George_Mather/Motion/* (2006)

[5] Podpora M., Computer vision in parallel computing - reducing communication time, *Proceedings of the 13th Conference Student EEICT 2007 vol.4,* (2007)

[6] Podpora M., O przyspieszaniu klastra sterującego robotem mobilnym, *Materiały konferencyjne Pierwszych Warsztatów Środowiskowych Doktorantów Politechniki Opolskiej*, Jarnołtówek, (2007), *in press*

***Author**: mgr inż. Michał Podpora, Opole University of Technology, WEAII I-1, Poland, 45-272 Opole, ul. Sosnkowskiego 31, ravyr@klub.chip.pl*