Biologically reasoned point-of-interest image compression for mobile robots

Michał Podpora, *Student Member, IEEE, Jan Sadecki*
Opole University of Technology

***Abstract* — In this article author describes image compression based on the idea of biological "yellow spot" – the quality/resolution is variable, depending on the distance from point-of-interest. Reducing the amount of data in robot's vision system enables possibilities of using a computer cluster for non-time-critical "mental" processing tasks like "memories" or "associations". This approach might be especially useful in HTM-based processing of robot's vision system data.**

***Keywords* — active vision, distributed computing, image coding, machine vision, robot vision systems**

I. Introduction

TODAY'S mobile robots are mostly controlled by embedded systems. A lightweight but efficient processing solution, powerful enough to acquire input data and to control mechanics can be bought for a fair price. On the other hand "intelligence" and behavior algorithms of a robot are much more demanding tasks. Using a vision system can also be a reason for replacing the processor with a faster one. However, powerful computers need more energy, which causes that the "mobility" of a robot is hindered by heavy batteries.

Few years ago, a few groups of scientists tried to make use of a computer cluster for processing of algorithms not demanding real-time processing. Computer clusters' efficiency was strongly affected by communication time – video data sent between nodes was causing serious performance problems. Since then, processors evolved, but wireless networks, which could connect mobile robot with the cluster, didn't seem to keep up with processors' evolution.

Designing a mobile robot's human interaction algorithm is always a great challenge, and a computer cluster would be a perfect solution for "mental" non-real-time processing tasks like "memories" or "associations". Unfortunately, using a computer cluster for image processing and object recognition (i.e. the usual way of making the robot understand the image) is not possible due to the communication time. Reducing the amount of the transferred data usually causes loss of information in images. Object recognition gives poor results on noisy images.

A. Biologically reasoned

A human eye, unlike "artificial" vision systems, does not acquire the whole image using the same detail level or resolution for every part of the image. In fact, greater part of "visual data" comes from a yellow spot – part of a retina containing fovea (i.e. small pit responsible for central vision with more closely packed cones) providing the sharpest, and the most detailed image [1]. Nature has found its way to reduce the amount of data transferred to the "supercomputer".

This approach to visual data compression (presented in Fig.1) causes fundamental change in object recognition.

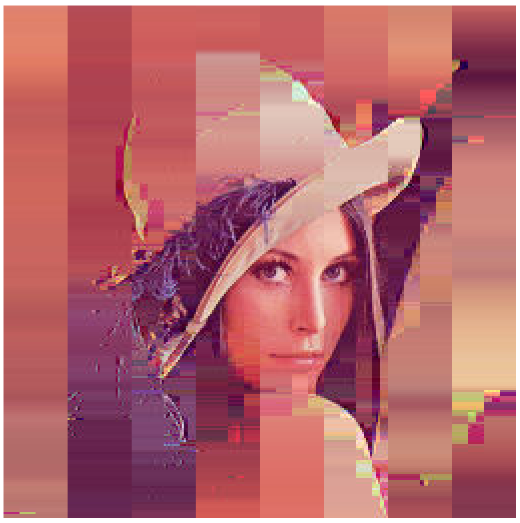
Fig. 1. Lena, vision system's yellow spot is fixed on her eye.

II. Object Recognition

An object recognition algorithm based on a sequence of images compressed like in Fig. 1, can be implemented using HTM networks (Hierarchical Temporal Memory). In HTMs it is absolutely natural to use spatial fragments of input data and to send it to the input of the network in a time sequence [2]. The idea of HTM network's understanding of the input data/signal [3] is based upon human brain's neocortex. A human eye makes very fast movements called saccades. These moves change the fixation point of retina's yellow spot causing change of observed detail. By generating sequence of details, it is possible to build up a virtual map of corresponding image.

A robot can also be programmed to use it's vision system this way. A video camera or a stereovision system acquires video data and subsequently, images are compressed with the use of yellow spot's coordinates for defining the point-of-interest (point of maximum quality of the image after compression). The information is passed to the HTM network on a remote computer and processed in a usual manner, just like any other information within HTM networks, making inference possible.

An abstract notion of a dog can be described as a sum of features like specific sounds or spatial patterns acquired by the retina (e.g. tail, nose, ear, paw) [4].

III. Yellow Spot

A. *Human eye*

In a biological eye, the quality/resolution of image is decreasing with the distance from the fixation point. The difference is not equal in every direction from the fixation point, although it seems to be so. If we look at the middle of a circle, the circle seems to be round (well, it is) but it is not transferred through the optic nerve as a circle (cones are not arranged uniformly on the retina) – although, the brain interprets it as a circle. This example shows that our “vision system” also makes some conversions and simplifications to the acquired data before processing.

B. *“Robot’s yellow spot”*

An “artificial” vision system acquires visual data as a predefined pattern of pixels organized in columns and rows. This is very problematic for people willing to experiment with a HTM network, because the network expects temporal sequences of spatial patterns on its input. The most frequent solution is to scan the whole image with the input sensor matrix.

Having the yellow spot in mind, “temporal sequence” can be understood not as a horizontal shift of the sensor matrix, but as a sequence of patterns found in the neighborhood of the yellow spot. Fig. 1 shows an idea of reducing the amount of data in the image, not loosing the quality or resolution of the “virtual yellow spot’s” coordinates neighborhood.

The image reduction cannot be made as it is shown in Fig. 2a, because the information outside the central area is sometimes also useful. E.g. HTM cannot decide to change yellow spot’s coordinates to the nose, because there is no nose in the Fig. 2a. Of course, well-trained HTM could “suppose” where the nose should be found, but at first, it should have good opportunities to learn.

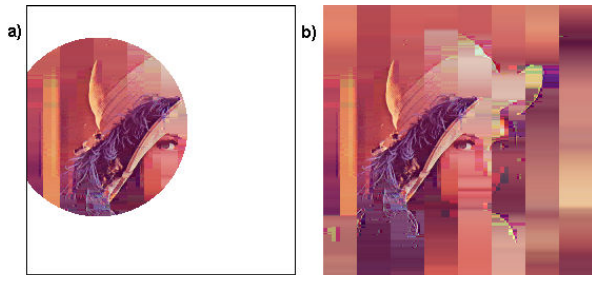

Fig. 2. Lena, vision system’s yellow spot is fixed on a detail. (a) The part of the image with the highest compression level (i.e. the lowest quality) had been cut-off. (b) The same image without cutting.

IV. Compression

It doesn’t matter what kind of compression algorithm is to be used for compressing the images exactly, but this algorithm should give following possibilities:

- the quality/resolution is changing smoothly with the distance from yellow spot’s coordinates,
- it would be very helpful, if the high-level compressed area would show (despite compression) existence of details.

V. Practical Realization

In the practical realization, the simplest possible DWT (discrete wavelet transform) – Haar wavelet transform – was used [5], analyzing horizontal details only. Afterwards, all calculated differential matrixes are compressed with simplified RLE (run-length encoding) procedure (only zeros are compressed). The quality of image was defined by the threshold level defined not as a constant value, but as a modified gaussian function (see Fig. 3).

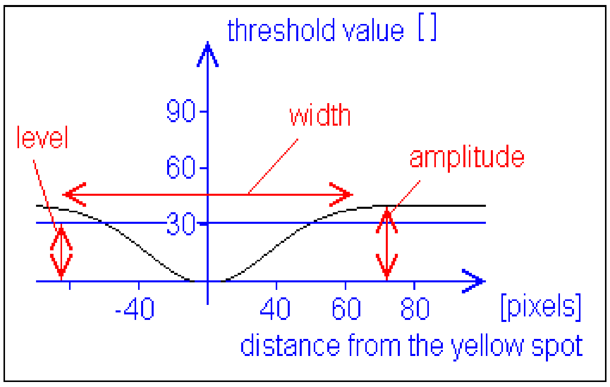

Fig. 3. Threshold function.

By modifying parameters presented on Fig. 3, it is possible to control parameters of the threshold function (see Fig.4).

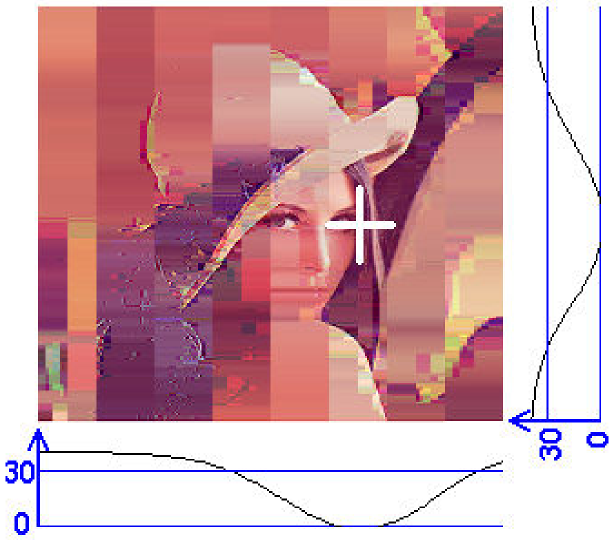

Fig. 4. Lena and threshold curves used within DWT transformation.

Parameters of the threshold function (i.e. the quality of the compressed image) can be chosen automatically (by comparing original image to the compressed one) or manually. In the latter case, the application developer has the opportunity to see the influence of a specified parameter on the compressed image.

Fig. 5 shows contour lines of the threshold function.

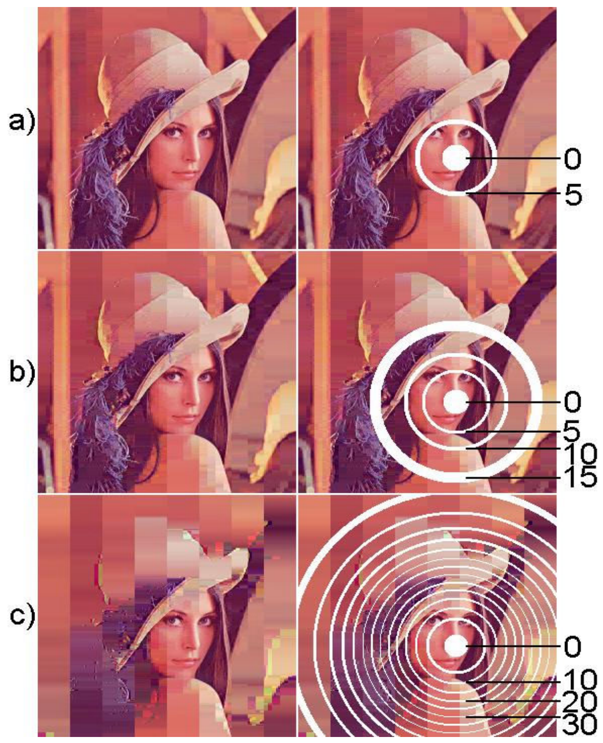

Fig. 5. Contour lines of the threshold function. White lines denote lines/areas where threshold modulo 5 = 0. Numbers describe threshold value in specified area.

The image in Fig. 5 c seems to be useless for image processing and object recognition algorithms, but the neighborhood of the yellow spot is compressed with the best quality. Therefore, in some cases (e.g. for HTM networks) such image is equally as useful as uncompressed one. It is good enough for analysis of the observed detail and for perceiving the existence of other details in its neighborhood.

TABLE 1: DATA SIZE OF AN EXAMPLE TRANSFORMED IMAGE

Image quality	*Example data size*
Best	e.g. 212 327 bytes
Very good (Fig. 5 a)	e.g. 105 261 bytes
Good (Fig. 5 b)	e.g. 72 070 bytes
Poor (Fig. 5 c)	e.g. 33 285 bytes

Values in table 1 correspond to images on Fig. 5. Although the original file contained 196662 bytes (196608 and header), first row of the table shows weaknesses of the RLE encoding.

The compressed image data size depends on threshold function parameters, source image complexity, exact coordinates of the yellow spot, and on other factors, therefore it is difficult to estimate it.

Entropy coding might produce better results [6] than proposed compression – it was not author's intention to find an ultimate compression algorithm, but rather to show usefulness of compressing images with variable quality/ /resolution.

VI. ACTIVE VISION VS YELLOW SPOT

Choosing a smaller area of the image to speed-up the processing is well-known idea, called "active vision". However, active vision usually ignores the rest of the image, while the proposed algorithm still gives the possibility to "notice" the existence of a new point-of-interest beyond the analyzed area. A DWT-transformed image reduces the information about small input signal changes and stores the information about big changes. Terms "big" and "small" are polarized by the threshold parameter (or function). This allows detecting contrasting (differencing in color or lighting) objects located anywhere within the field of view (or detecting movement) while offering significant data size reduction. The developed application's algorithm is designed to transform the image using threshold function with predefined maximum value of 32. It offers much better image quality than in Fig. 5c and more efficient data compression than Fig. 5b. In Fig. 5c and in Fig. 5b the maximum threshold value is 18 and 68, respectively. The main goal of using the threshold function instead of a fixed threshold value is to compress the image not loosing the quality in the neighborhood of yellow spot, and not loosing the information about contrasting or moving objects.

Compression of images is essential for reducing one of the computer cluster's parameters – communication time. Using a computer cluster with raw data is pointless.

Although lossy compression often makes further image processing very difficult or even impossible, the proposed algorithm is transparent for a HTM network, because HTM networks require only a specified spatial fragment of the image. Even the image shown in Fig. 5c should be sufficient for a HTM network, because the yellow spot's neighborhood is compressed with maximum quality.

VII. INFERENCE

HTM networks consist of nodes connected together in a tree topology. The lowest layer of nodes uses the sensory data, whereas all other layers usually use preprocessed knowledge from a previous layer. The HTMs' hierarchical structure combined with nodes' algorithm (based on autoassociative memory) gives the possibility to analyze the input data as a spatial and/or temporal sequence of input patterns – just like it happens in neocortex. [3]

HTMs are capable of discovering causes in the analyzed input data, and inferring causes of novel input. Optionally they can be also used for predicting the input data or controlling the application's / robot's behavior. [2]

Discovering causes is possible, because objects are not stored in memory as spatial patterns. Although the input data is acquired as a spatio-temporal pattern of pixels, this representation can be found only in the first, sensory-level layer. All other layers build an abstract pattern of active nodes to represent a specified object. An object in the real world is called "cause" and its representation in HTM (a probability of each of the learned causes) is called "belief". As an example, a spoken language can be used. Each layer of a HTM contains some active and some inactive nodes, while every layer represents more and more abstract and complex "causes". The sensory data carries the information about current "active" sound frequencies, but higher layers can indicate phonemes, words, phrases, sentences, and ideas. This process is similar for any possible input data, so implementing HTM's in a machine vision approach should not induce any change of the inferring algorithm.

HTMs can be easily used for image recognition [7], if the input node is modified to use spatial, 2-dimensional input data. The temporal aspect of sensory data can be generated artificially, by scanning whole image line-by-line with the input nodes' input arrays (it is a very popular approach). It can also be derived from biology – just like human eye moves from one fixation point to another with every saccade [6].

HTM's layers are built of nodes, where each node learns "causes" and forms "beliefs" [2]. Nodes in the first layer take arrays of pixels and form "beliefs" about e.g. basic shapes (see Fig. 6, first column). Next layer nodes analyze spatial and/or temporal coexistence of active input layer nodes and form further "beliefs" (Fig. 6, columns 2-5).

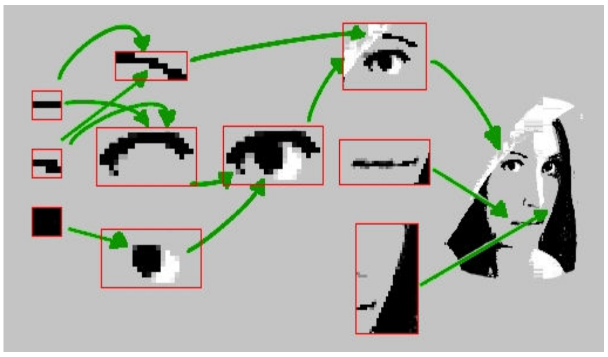
Fig. 6. Hierarchy of patterns on different abstraction levels. The face on the right side of this figure should not be understood as an assembled bitmap, but rather as an abstract state of coexistence of "beliefs" from the previous layer of nodes (e.g. "an eye" [90% of a certainty] +"a mouth" [40% of a certainty] +"a nose" [40% of a certainty] = "a face" [50% of a certainty])

Various parts of the image can be acquired separately by moving the yellow spot from one part to another. The machine vision system must be cognizant of analyzing one particular object. A highly probable belief of seeing an eye followed (after a saccade) by a highly probable belief of seeing a nose might state that the robot is currently looking at a human being. This might be checked e.g. by an intentional saccade to another fixation point.

As shown in Fig. 7, use of saccades and fixation points together with the described DWT+RLE algorithm is a very powerful idea.

The most important feature of this solution is that it uses HTMs – pixels located in the neighborhood of every fixation point are used for generating "beliefs" about fragments of an image (e.g. "an eye [80% of a certainty]", "a mouth [60% of a certainty]"). These "beliefs" are crucial for inferring what exact "causes" are reasons of a final "belief".

While HTMs are based on autoassociative memory, input data can be incomplete (or slightly differ from the learned example) and the algorithm would still be able to recognize it.

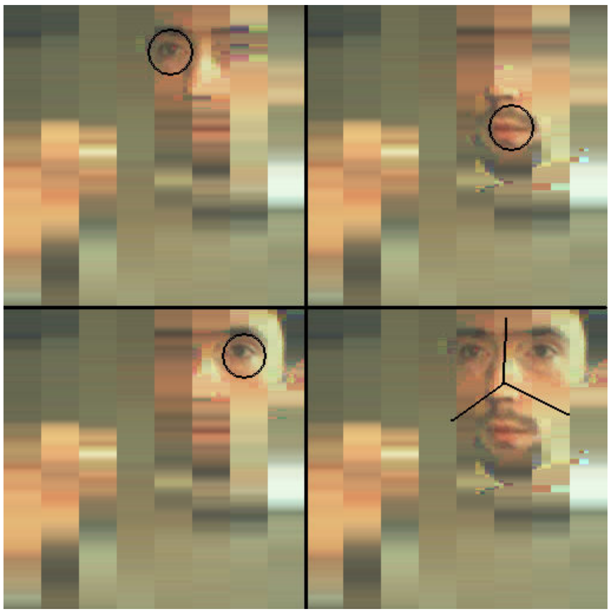
Fig. 7. Three fixation points of the acquired image seem to be enough for a human eye to recognize a face. Although human brain has already recognized the object, saccades still occur to send more details about it.

A house with a blue roof will be recognized as a house (even if the robot have never "seen" a blue roof before), because "beliefs" of some of nodes indicate that this object has windows, door, etc. and the most probable "cause" of the final "belief" (among all known "causes") is "a house".

Indicating if the analyzed object is already known or new also seems to be easy task. For all known objects, the belief distribution is peaked, and for unknown objects, it is flat. [2]

VIII. Election of a Next Fixation Point

The developed application has two modes of defining a new fixation point. The first one (with lower priority) defines new fixation point in the middle of a moving object and starts analyzing it. The second operating mode is active only when an object is being analyzed – other objects' movement is ignored, and further fixation points are defined depending on decision of the analyzing algorithm.

Current version of the application is not using HTMs' prediction feature, which seems to be the most promising direction of research.

While HTMs store the information about spatio-temporal sequence of patterns, it is possible to make some predictions. If the analyzed image contains “an eye” and “a mouth”, some of the nodes could already indicate, that it is probable that the common “cause” is “a face” (Fig.8). However, there are more details describing “a face” (e.g. “a nose” or “an eyebrow”). HTM might use this information for checking if there is “a nose” (see Fig.8). Choosing next fixation point with the use of HTMs is a complex task, but it seems to be the most resolute and effective way.

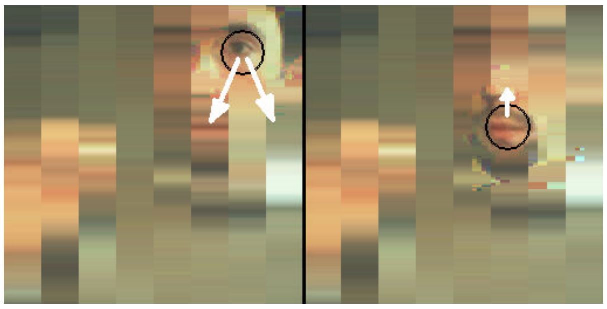
Fig. 8. Election of another fixation point, basing on HTM prediction feature (predicting spatio-temporal patterns of partially recognized objects).

IX. Mobile Robots and Computer Clusters

The most problematic task in using computer cluster as an enhancement of robot’s memory and/or association ability was the communication time. Input data of robot’s vision system (i.e. images from its camera) produced extremely high volume of the traffic. No parallel algorithm could manage it.

The presented idea of yellow spot enables the possibility of using the cluster. The parallel algorithm should be designed to use task parallelism rather then data parallelism [8]. Analysis of half of an image is pointless, although some data parallelism can also be considered (depending on the case of use).

Of course, all real-time reactions of mobile robot should be implemented locally on the embedded computer, because of computer cluster’s huge delay (regardless of the image compression algorithm).

A computer cluster is, in general, a group of loosely coupled computers that work together closely so that they can be viewed as though they are a single computer [9]. The components of a cluster are commonly connected to each other through fast local area networks. Clusters are usually deployed to improve performance and/or availability over that provided by a single computer, while typically being much more cost-effective than single computers of comparable speed or availability. Efficiency of parallel (cluster) implementation of computation depends, in general, on the processor performance and communication bandwidth of the cluster. The value of communication bandwidth is especially important for the problems for which exchange of data between each pair of the cluster nodes is required, especially for systems containing very large number of processors. Highly parallel computers contain thousands of processors. If, for a given problem size, communication will eventually dominate computation as the number of processors is increased, then the speedup cannot scale with large numbers of processors without introducing additional levels of parallelism.

Computational burdens encountered when solving complex problems of control of multidimensional processes (robots), can be essentially reduced by decomposing the problem into a number of subproblems and by solving a set of subtasks associated with a certain coordination task.

Assuming a set of all operations required to be carried out for solving one local task as the least portion of a task which can be performed by a processor, the discussed two-level algorithm can in a simple manner be implemented parallel in multiprocessor system. Notice that the Master-Slave structure, being natural for methods of this type, is less effective for cluster systems with a large number of processors

Furthermore, in the distributed memory systems, each of the tasks specified in a given algorithm, should be solved in a parallel way, i.e. in case of the considered methods the coordination algorithm ought also to be solved parallel. The vertical communication corresponds in principle to data exchange between two algorithms implemented by the same processor, namely, between the algorithm implementing a local task and a fragment of, allocated to this processor, the coordination algorithm implemented parallel. On the other hand, the horizontal communication corresponds to that one between particular processors, including the exchange of data necessary for the correct implementation of the coordination algorithm.

Decreasing of the computation time arising as the consequence of parallel realization of two-level optimization problem is rather evident. But decomposition can lead to decrease of the communication requirements too, especially, for the each-to-each communication problem (bi-directional data transmission between each pair of processors). For example, a number of required communication tasks L_s for the latter problem is determined as:

$$L_S = P(P-1), \qquad (1)$$

where P denotes the number of processors.

If we divide all processors into L groups (nodes), each of them containing approximately P/L processors, the total number of communication tasks required to perform the same exchange of data, is determined as follows:

$$L_d = \frac{P}{L}\left(\frac{P}{L}-1\right)L + L(L-1) + \left(\frac{P}{L}-1\right)L, \qquad (2)$$

where $(P/L)(P/L\text{-}1)L$ denotes the number of required communications task in all groups of processors (nodes), $L(L-1)$ denotes the number of communication tasks between all groups and $(P/L-1)L$ denotes total number of communication tasks allowing to send data obtained from other groups into all processors in each group.

Fig. 9 presents values of the factor $S_d=L_s/L_d$ as a function of a number of processor groups L for given value of P. This picture shows that the communication requirements can be significantly decreased as a consequence of the realized decomposition.

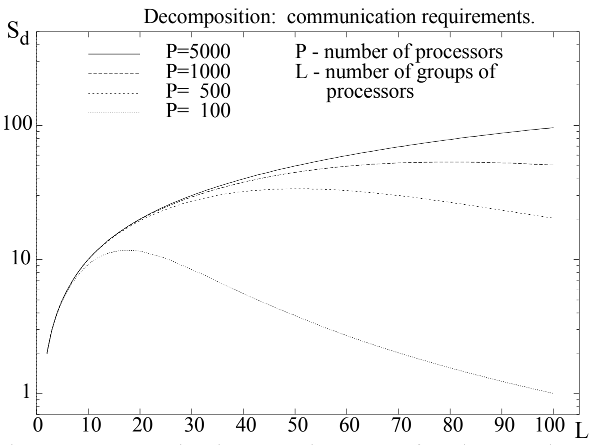

Fig. 9. Communication requirements for the two-level parallel each-to-each communication.

X. Conclusion and Future Work

Analysis of high-resolution digital images demands computation power and operating memory. Analysis of image sequences makes it even more challenging. Using a computer cluster Not only proper data representation on system input is an important aspect of fast and efficient processing, but also defining proper "neural" architecture [10] and problem decomposition.

HTM networks seem to be the most natural conception in vision understanding, and a new hope for active vision adherents. While HTM networks are relatively new idea of implementing memory and vision understanding, it is very hard to predict efficiency, elasticity and robustness of developed system before it is ready.

DWT-targeted RLE compression, described in this paper, is already implemented in a parallel application (see Fig.10). HTM network is known to manage with image recognition [7], but it is not fully implemented in the same application yet. The next step is to implement a HTM network in an example vision system consisting of mobile robot with video capturing devices, and a computer cluster for advanced analysis of image, object recognition and inferring (HTM-based vision understanding).

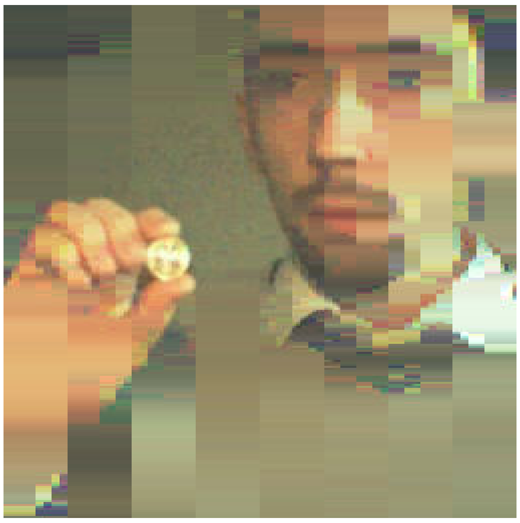

Fig. 10. A single frame with yellow spot fixed on the coin is firstly DWT-transformed. Secondly, the frame is RLE-compressed, then transferred from the input node to the computer cluster and finally IDWT-transformed for further processing. After processing, a new fixation point coordinates are sent back to the input node.

References

[1] E. Hecht, *Optics*. Addison Wesley, 4th edition, 2002.
[2] J. Hawkins, G. Dileep, *Hierarchical temporal memory – concepts, theory and terminology*. Numenta Inc., 2007. Available: http://www.numenta.com/Numenta_HTM_Concepts.pdf
[3] J. Hawkins, S. Blakeslee, *On intelligence*. Times Books, 2004.
[4] J. Hawkins, "Learn like a human," *IEEE Spectrum*, vol.4, April 2007.
[5] S. Mallat, *A wavelet tour of signal processing*. Academic Press, 1999.
[6] M. Podpora, *Biologically reasoned machine vision: RLE vs. entropy-coding compression of DWT-transformed images*. Proceedings of the 14th Conference Student EEICT 2008, Brno, 2008.
[7] Numenta Inc., *Numenta Pictures Demonstration Program*. 2007. Available: http://www.numenta.com/about-numenta/technology/ /pictures-demo.php
[8] J. Sadecki, *Parallel optimization algorithms and investigation of their efficiency: parallel distributed memory systems.* Opole University of Technology, Opole 2001, (in Polish).
[9] M. Baker, *Cluster Computing White Paper*. 2001. Available: http://arxiv.org/abs/cs/0004014/
[10] M. Nałęcz, W. Duch, J. Korbicz, L. Rutkowski, R. Tadeusiewicz, *Biocybernetics and biomedical engineering 2000 – vol 6 –neural networks*. Akademicka Oficyna Wydawnicza Exit, Warszawa 2000, (in Polish).